ORCA Manual

Release 6.1

Max-Planck-Institut für Kohlenforschung

Jul 17, 2025

PREFACE

ORCA 6.1 Foreword i

ORCA 6.1 Highlights iii

How to cite v

How to use this manual ix

1 Quickstart Guide 1
- 1.1 About ORCA . . . 1
- 1.2 Installation . . . 1
- 1.3 Graphical User Interfaces (GUI) . . . 3
- 1.4 Running ORCA . . . 3
- 1.5 Hello water! Your first ORCA calculation . . . 4
- 1.6 General Recommendations . . . 6
- 1.7 Troubleshooting Problems in ORCA . . . 10

2 Essential Calculation Elements 23
- 2.1 General Structure of the Input File . . . 23
- 2.2 Input of Coordinates . . . 31
- 2.3 Basic Calculation Settings . . . 37
- 2.4 Control of Output . . . 38
- 2.5 Parallel and Multi-Process Runs . . . 42
- 2.6 Self-Consistent-Field (SCF) . . . 47
- 2.7 Basis Sets . . . 60
- 2.8 Resolution-of-the-Identity (RI) . . . 94
- 2.9 Numerical Integration . . . 103
- 2.10 Details on the Numerical Integration Grids . . . 104
- 2.11 Counterpoise Corrections . . . 108
- 2.12 Relativistic Calculations . . . 115
- 2.13 Implicit Solvation . . . 126
- 2.14 Integral Handling . . . 151
- 2.15 The SHARK Integral Package and Task Driver . . . 163
- 2.16 SCF Stability Analysis . . . 167
- 2.17 Finite Electric Fields . . . 169
- 2.18 Fragment Specification . . . 172
- 2.19 ORCA and Symmetry . . . 191
- 2.20 Choice of Initial Guess and Restart of SCF Calculations . . . 197
- 2.21 Frozen Core Options . . . 201
- 2.22 CP-SCF Options . . . 203
- 2.23 Numerical Gradients . . . 205

3 Model Chemistries 207
- 3.1 Wavefunction Types: RHF/RKS, UHF/UKS, ROKS and more . . . 207
- 3.2 Hartree Fock Theory . . . 211

3.3 Density Functional Theory (DFT) . . . 211
3.4 Dispersion Corrections . . . 235
3.5 Semiempirical Methods . . . 248
3.6 Composite Methods (3c methods) . . . 258
3.7 Analytic Density Functional Theory (ADFT) . . . 262
3.8 Random Phase Approximation (RPA) . . . 265
3.9 Perturbation Theory - MP2 . . . 266
3.10 Coupled Cluster and CI Theories (MDCI) . . . 292
3.11 Correlated Methods using Automatic Code Generation (AUTOCI) . . . 342
3.12 Arbitrary Order Coupled-Cluster (MRCC interface) . . . 357
3.13 Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS) . . . 358
3.14 Approximate Full CI Calculations in Subspace: ICE-CI . . . 446
3.15 Density Matrix Renormalization Group (DMRG) . . . 464
3.16 N-Electron Valence State Perturbation Theory (NEVPT2) . . . 472
3.17 Complete Active Space Peturbation Theory (CASPT2 and CASPT2-K) . . . 490
3.18 CASSCF and DFT . . . 494
3.19 Multireference Configuration Interaction and Pertubation Theory (uncontracted) . . . 499
3.20 Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC) . . . 557
3.21 Dynamic Correlation Dressed CAS . . . 584
3.22 Full Configuration Interaction . . . 589
3.23 Molecular Mechanics . . . 593

4 Structure and Reactivity **601**
4.1 Geometry Optimizations . . . 601
4.2 Surface Scans . . . 620
4.3 Transition State Searches . . . 623
4.4 Intrinsic Reaction Coordinate . . . 631
4.5 Nudged Elastic Band Method . . . 634
4.6 Nudged Elastic Band Method . . . 637
4.7 Vibrational Frequencies . . . 654
4.8 Thermochemistry . . . 659
4.9 Conical Intersections . . . 667
4.10 Minimum Energy Crossing Points (MECP) . . . 670
4.11 GOAT: global geometry optimization and ensemble generator . . . 673
4.12 SOLVATOR: Automated Explicit Solvation . . . 687
4.13 DOCKER: Automated Docking Algorithm . . . 696

5 Spectroscopy and Properties **709**
5.1 Population Analysis . . . 709
5.2 Natural Bond Orbital (NBO) Analysis . . . 725
5.3 Fractional Occupation Number Weighted Density (FOD) . . . 737
5.4 Excited States Calculations . . . 742
5.5 Excited State Dynamics . . . 764
5.6 Excited States via RPA, CIS, TD-DFT and SF-TDA . . . 809
5.7 Excited States via ROCIS and ROCIS/DFT . . . 854
5.8 Excited States via MCRPA . . . 878
5.9 Excited States via EOM-CCSD . . . 886
5.10 Excited States via STEOM-CCSD . . . 895
5.11 Excited States via IH-FSMR-CCSD . . . 903
5.12 Excited States via PNO-based coupled cluster . . . 906
5.13 Excited States via DLPNO-STEOM-CCSD . . . 908
5.14 One Photon Spectroscopy . . . 909
5.15 Core-Level Spectroscopy with Coupled Cluster Methods . . . 918
5.16 Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program . . . 933
5.17 Absorption and Fluorescence Bandshapes using `ORCA_ASA` . . . 978
5.18 Vibrational Spectroscopy . . . 986
5.19 Anharmonic Analysis and Vibrational Corrections using VPT2/GVPT2 and `orca_vpt2` . . . 1000

5.20 Electrical Properties - Electric Moments and Polarizabilities . . . 1006
5.21 Nuclear Magnetic Resonance (NMR) Parameters . . . 1016
5.22 Paramagnetic NMR Shielding Tensors . . . 1029
5.23 Spin-rotation Constants . . . 1030
5.24 Electron Paramagnetic Resonance (EPR) Parameters . . . 1031
5.25 MP2 Level Magnetic Properties . . . 1045
5.26 CASSCF Linear Response . . . 1047
5.27 Mössbauer Parameters . . . 1051
5.28 The Spin-Orbit Coupling Operator . . . 1054
5.29 Broken-Symmetry Wavefunctions and Exchange Couplings . . . 1058
5.30 Decomposition of the Magnetic Exchange Coupling . . . 1060
5.31 Magnetic Properties Through Quasi Degenerate Perturbation Theory . . . 1062
5.32 Magnetic Relaxation . . . 1072
5.33 Simulation of (Magnetic) Circular Dichroism and Absorption Spectra . . . 1078
5.34 Interface to SINGLE_ANISO Module . . . 1082
5.35 Interface to POLY_ANISO Module . . . 1100
5.36 DeltaSCF . . . 1111
5.37 Extended Transition State with Natural Orbitals for Chemical Valence (ETS-NOCV/EDA-NOCV) 1125
5.38 Local Energy Decomposition . . . 1134
5.39 The Hartree-Fock plus London Dispersion (HFLD) Method . . . 1158
5.40 ADLD(D): Atomic Decomposition of DFT-D London Dispersion Energy . . . 1161
5.41 Static Ground State DFT (SGS-DFT) . . . 1163

6 Multiscale Simulations 1177
6.1 Overview on ORCA's Multiscale Implementation . . . 1177
6.2 Multiscale Simulations of molecular systems . . . 1189
6.3 CRYSTAL-QMMM . . . 1196
6.4 QM/MM via Interfaces to ORCA . . . 1201
6.5 Fast Multipole Method . . . 1206

7 Molecular Dynamics 1213
7.1 *Ab initio* Molecular Dynamics . . . 1213

8 Workflows and Automatization 1243
8.1 ORCA Python Interface (OPI) . . . 1243
8.2 Compound . . . 1243
8.3 More Details on Compound . . . 1250
8.4 Compound Examples . . . 1315

9 Utilities and Visualization 1339
9.1 Orbital and Density Plots . . . 1339
9.2 Utility Programs . . . 1348
9.3 orca_2json . . . 1416
9.4 Property File . . . 1436

10 Architecture of ORCA 1507
10.1 ORCA Structure . . . 1507
10.2 The Architecture of ORCA . . . 1508
10.3 Units and Conversion Factors . . . 1513

A Detailed change log 1515
A.1 Changes ORCA 6.1 . . . 1515
A.2 Changes ORCA 6.0.1 . . . 1527
A.3 Changes ORCA 6.0.0 . . . 1533

B Publications Related to ORCA 1537
B.1 Method development . . . 1537
B.2 Relevant Applications, Benchmarks and Reviews . . . 1543
B.3 Classification by Topic . . . 1549

B.4 Applications that make use of or include the following 1566
B.5 Reviews of interest 1574

C Glossary **1577**

Bibliography **1583**

Index **1623**

- An *ab initio*, DFT and semiempirical SCF-MO package -

Version 6.1

Design and Scientific Directorship: **Frank Neese**
Technical Directorship: **Frank Wennmohs**

Max-Planck-Institut für Kohlenforschung
Kaiser-Wilhelm-Platz 1, 45470 Mülheim a. d. Ruhr, Germany
department-mts@kofo.mpg.de

With contributions from:

MPI für Kohlenforschung:
Ahmet Altun, Daniel Aravena, Michael Atanasov, Alexander A. Auer, Ute Becker, Giovanni Bistoni, Markus Bursch, Dmytro Bykov, Marcos Casanova-Páez, Vijay G. Chilkuri, Pauline Colinet, Dipayan Datta, Achintya Kumar Dutta, Nicolas Foglia, Dmitry Ganyushin, Miquel Garcia-Rates, Tiago L. C. Gouveia, Yang Guo, Andreas Hansen, Ingolf Harden, Benjamin Helmich-Paris, Lee Huntington, Róbert Izsák, Riya Kayal, Emily Kempfer, Christian Kollmar, Axel Koslowski, Simone Kossmann, Lucas Lang, Marvin Lechner, Spencer Leger, Dagmar Lenk, Dimitrios G. Liakos, Dimitrios Manganas, Dimitrios A. Pantazis, Anastasios Papadopoulos, Taras Petrenko, Peter Pinski, Shashank V. Rao, Christoph Reimann, Marius Retegan, Christoph Riplinger, Michael Roemelt, Masaaki Saitow, Barbara Sandhöfer, Yorick L. A. Schmerwitz, Kantharuban Sivalingam, Bernardo de Souza, Georgi L. Stoychev, Van Anh Tran, Willem Van den Heuvel, Zikuan Wang, Hang Xu

FACCTs GmbH:
Markus Bursch, Nicolas Foglia, Miquel Garcia-Rates, Ingolf Harden, Hagen Neugebauer, Mustafa C. Ozdemir, Anastasios Papadopoulos, Christoph Plett, Christoph Riplinger, Bernardo de Souza, Georgi L. Stoychev

Other institutions:
Vilhjálmur Åsgeirsson, Christoph Bannwarth, Giovanni Bistoni, Martin Brehm, Ronald Cardenas, Garnet Chan, Martina Colucci, Sebastian Ehlert, Marvin Friede, Lars Goerigk, Stefan Grimme, Waldemar Hujo, Mihály Kállay, Holger Kruse, Marcel Müller, Jiri Pittner, Philipp Pracht, Gianluca Regni, Liviu Ungur, Edward Valeev, Lukas Wittmann

Additional contributions to the manual from:
Wolfgang Schneider, Gregor Giesen, Johannes Grün

ORCA 6.1 FOREWORD

It was less than one year ago, July 2024, that ORCA 6.0 was released to the general public by a proud and happy but also thoroughly exhausted development team. As we pointed out at the time, ORCA 6.0 was less an update to an existing program but rather a completely re-imagined quantum chemistry program suite in which large parts of the massive code base where re-designed and rewritten from scratch. This process took nearly three years and resulted in a program that was much leaner, much cleaner and as a result also more modern and more efficient.

Our promise at the time of the release of ORCA 6.0 was that the new infrastructure would allow for much more rapid and more confident method development than ever before. With the release of ORCA 6.1, we believe that we made good on this promise. Despite less than one year has passed since the release of ORCA 6.0, version 6.1 is packed with new features and further significant improvements under the hood. This release can therefore indeed be regarded as a proof-of-principle for the new infrastructure. It turned out to be so encouraging that we have every intention to keep up the pace and aim at a release of a new ORCA version approximately every 12 months – hold us to it!

ORCA 6.1 is the result of the work of many individuals. The bulk of the work has been done by the development teams at the Max Planck institute für Kohlenforschung in Mülheim an der Ruhr, Germany and at FACCTs in Cologne, Germany. However, we also very much enjoy our increasingly numerous collaborations with other research groups around the world - thank you so much, you rock!

I want to express my gratitude to all the many people who have put their enthusiasm and hard work into making this release reality. All of them were, are and will continue to be instrumental for the success of what now has become a very large project! An ORCA release would not be possible without the ORCA development team, led by Frank Wennmohs and featuring Ute Becker, Dagmar Lenk, Dimitrios Liakos, Kantharuban Sivalingam and Axel Koslowski, who have done invaluable work in making sure that the code stays healthy, our libraries are up to date, that it compiles on all platforms (*and* give correct results everywhere), educate the developers on good programming style, fixing countless bugs, parallelizing everything and going after each and every deviation that our massive test suite reports.

Alexander Auer and Markus Bursch have taken the lead on the important but incredibly tedious task to restructure the mammoth that the ORCA manual has become, making sure that it is up to date and available in HTML and PDF versions. Thank you so much! The importance of that work cannot possibly be overemphasized.

It gives me enormous pleasure to observe the increasing impact on industrial computational chemistry that FACCTs has under the outstanding leadership of Christoph Riplinger. This not only leads to rapid and sustainable growth of the company but also provides many important ideas and incentives for exciting science. The development team at FACCTs continues to make many important contributions to the code, exciting new features, new methods, new workflows together with an unwavering commitment to precision and excellence. I should specifically mention Bernardo de Souza, Georgi Stoychev, Miquel Garcia-Rates, Ingolf Harden, Nicolas Foglia and Hagen Neugebauer. Thank you all – it is a pleasure!

Working on an ORCA release is not a chore, it is a labour of passion, joy and dedication. It gives me incredible joy to work on something that proves to be useful for tens of thousands of people all around the globe. It also gives me incredible joy to work on this as part of a team that is enthusiastic, highly motivated, highly skilled and highly dedicated.

Now you, the ORCA community, holds the next chapter of our journey in your hands. We are proud of ORCA 6.1 and we promise that this won't be our last rodeo in the continuing ORCA development. We hope that you enjoy it as much as we enjoyed working on it - and - of course, we hope that it helps you solving chemical problems, large or small, whatever your scientific pathway may be.

Frank Neese

On behalf of the ORCA development team

May 26th, 2025

ORCA 6.1 HIGHLIGHTS

Based on the greatly improved infrastrucuture introduced with ORCA 6.0 last year, we were able to make rapid progress such that we can present ORCA 6.1 to you. ORCA 6.1 features further efficiency enhancements and also a host of new functionality that we are excited about :

- Extended support for magnetic property calculations at high-level ab initio methods
- Auto fragmentation: fully automated fragmentation algorithm
- Analytical computation of Raman intensities
- Simplified multiscale setup: automated generation of of QM and active region
- RESP charges: restrained electrostatic potential charges
- Updates on GOAT and DOCKER
- Electrostatic potentials: automatic ESPs via orca_plot

For a detailed change log of ORCA 6.1, please see the *detailed change log*

HOW TO CITE

As ORCA is, and will always, be free for academic use, the people behind ORCA rely on being cited for their hard work developing it and making it the useful tool it actually is. Therefore, we kindly ask you to not only cite the actual generic ORCA reference of the used version but also the individual works relevant to the methods used in your ORCA calculations. To identify these more easily, we added an automated printout to ORCA 6 that provides you with recommendations.

The generic reference for ORCA is:

Neese, F. *The ORCA program system Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, **2012**, *2*, *1*, 73–78 (DOI: 10.1002/wcms.81)

BibTeX:

```
@article{ORCA,
author = {Neese,F.},
title = {The ORCA program system},
journal = {Wiley Interdiscip. Rev. Comput. Mol. Sci.},
volume = {2},
number = {1},
pages = {73-78},
DOI = {10.1002/wcms.81},
year = {2012},
type = {journal Article}
}
```

The current update for ORCA 6 is:

Neese, F. *Software update: the ORCA program system – Version 6.0 Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, **2025**, *15*, *2*, e70019 (DOI: 10.1002/wcms.70019)

BibTeX:

```
@article{ORCA6,
author = {Neese, Frank},
title = {Software Update: The ORCA Program System—Version 6.0},
journal = {Wiley Interdiscip. Rev. Comput. Mol. Sci.},
volume = {15},
number = {2},
pages = {e70019},
keywords = {ab initio calculations, density functional theory, embedding methods,␣
↪global optimization, quantum chemistry},
doi = {10.1002/wcms.70019},
year = {2025}
}
```

Beside these ORCA provides you with a detailed list of recommended citations at the end of any ORCA run. For example for the input

```
! PBE CPCM(Water)

*XYZFILE 0 1 structure.xyz
```

the end of the output will look like:

```
--------------------------------
SUGGESTED CITATIONS FOR THIS RUN
--------------------------------
```

(continues on next page)

(continued from previous page)

```
Below you find a list of papers that are relevant to this ORCA run
We neither can nor want to force you to cite these papers, but we appreciate if␣
↪you do
You receive ORCA, which is the product of decades of hard work by many␣
↪enthusiastic individuals, for free
The only thing we kindly ask in return is that you cite our papers,
We deeply appreciate it, if you show your appreciation for ORCA by not just citing␣
↪the generic ORCA reference.

Please note that relegating all ORCA citations to the supporting information does␣
↪*not* help us.
SI sections are not indexed - citations you put there will not count into any␣
↪citation statistics
But we need these citations in order to attract the funding resources that allow␣
↪us to do what we are doing

Therefore, if you are a happy ORCA user, please consider citing a few of the␣
↪papers listed below in the main body of your paper

In addition to the list printed below, the program has created the file orca.
↪bibtex that contains the list in bibtex format
You can import this file easily into all common literature databanks and citation␣
↪aid programs


List of essential papers. We consider these as the minimum necessary citations

  1. Neese,F.
     Software update: the ORCA program system, version 6.0
     WIRES Comput. Molec. Sci., 2025 15(2) e70019
     doi.org/10.1002/wcms.70019

List of papers to cite with high priority. The work reported in these papers was␣
↪absolutely
necessary for this run to complete.
Our perspective: the developers of density functionals and basis sets usually get␣
↪cited in chemistry papers
Good! But without the algorithms to do something with them, the functionals or␣
↪basis sets would not do anything.
Hence, in our opinion, the algorithm design and method developments papers are␣
↪equally worthy of getting cited

  1. Neese,F.
     An improvement of the resolution of the identity approximation for the␣
↪formation of the Coulomb matrix
     J. Comp. Chem., 2003 24(14)1740-1747
     doi.org/10.1002/jcc.10318
  2. Garcia-Rates,M.; Neese,F.
     Effect of the Solute Cavity on the Solvation Energy and its Derivatives␣
↪within the Framework of the Gaussian Charge Scheme
     J. Comput. Chem., 2020 41 922-939
     doi.org/10.1002/jcc.26139
  3. Neese,F.
     The SHARK Integral Generation and Digestion System
     J. Comp. Chem., 2022  1-16
     doi.org/10.1002/jcc.26942

List of suggested additional citations. These are papers that are important in the
↪'surrounding' of
of this run, or papers that preceded the highly important papers. If you like your␣
↪results we are grateful for a citation.
```

(continues on next page)

(continued from previous page)

```
  1. Neese,F.
     The ORCA program system
     WIRES Comput. Molec. Sci., 2012 2(1)73-78
     doi.org/10.1002/wcms.81
  2. Neese,F.
     Software update: the ORCA program system, version 4.0
     WIRES Comput. Molec. Sci., 2018 8(1)1-6
     doi.org/10.1002/wcms.1327
  3. Neese,F.; Wennmohs,F.; Becker,U.; Riplinger,C.
     The ORCA quantum chemistry program package
     J. Chem. Phys., 2020 152 Art. No. L224108
     doi.org/10.1063/5.0004608

List of optional additional citations

  1. Neese,F.
     Approximate second-order SCF convergence for spin unrestricted wavefunctions
     Chem. Phys. Lett., 2000 325(1-3)93-98
     doi.org/10.1016/s0009-2614(00)00662-x
```

Here you see, that even a simple single point calculation in implicit water makes use of various methods that may not be obvious at first glance but may be cited. For example ORCA uses an improved CPCM treatment that should be cited as:

```
  3. Garcia-Rates,M.; Neese,F.
     Effect of the Solute Cavity on the Solvation Energy and its Derivatives␣
↪within the Framework of the Gaussian Charge Scheme
     J. Comput. Chem., 2020 41 922-939
     doi.org/10.1002/jcc.26139
```

For convenience, ORCA further provides a `.bibtex` file that contains BibTeX entries that can easily be exported to any reference manager or your LaTeX bibliography.

HOW TO USE THIS MANUAL

The ORCA manual has been growing for decades and all developers take great care to document and describe all features or ORCA as good as they can. From ORCA 6.0 to ORCA 6.1 this manual has undergone a major restructuring. What you are looking at has been and is a huge team effort and should meet the needs of absolute beginners with little theory background as well as the requirements of experienced developers.

Help us with your feedback

With the initial restructuring of the manual for ORCA 6.1, we made substantial changes that hopefully improve the readbility and general quality of the manual. However, this was only the beginning and we are glad to hear any constructive feedback from you via the ORCA forum on how to improve further future.

The new structure starts with a *quickstart guide* and a *guideline for installation* that should get everyone started. The overall structure is ordered such that we first summarize the *essential elements of ORCA* that a user needs to know when perfoming standard calcuations like *how to structure the input*, which *basis sets* and algorithms are available etc. Next, we outline the available *electronic structure approaches* that can be used to compute energies, structures and properties ranging from *Hartree-Fock* and *DFT* to high level ab-initio methods. The next chapter focuses on *structure and reactivity* and includes all approaches to optimize *geometries*, find *transition states* and compute relative (free) energies in various settings. *Molecular spectroscopy and properties* includes everything that can be computed once a structure has been established and ranges from density and wavefunction analysis to ground- and excited state properties up to complex higher order properties like, for example, *MCD*. The last few sections deal with important features for complex tasks like running *molecular dynamics*, *multiscalar approaches* or orchestrating *automated workflows with ORCA* using, e.g., the *compound* script language. Last but not least we have notes on important [utilities] that can be used for post-processing and some technical notes on the *ORCA architecture*.

Most sections will start with a short outline of the theory with references and some simple examples to get you started, further details like keyword lists and advanced examples are typically found later on in the sections.

We hope that you will find the manual useful and would be very happy if you give feedback online in the ORCA forum.

CHAPTER

ONE

QUICKSTART GUIDE

1.1 About ORCA

ORCA[1] is a general-purpose quantum chemistry package initiated by Frank Neese in 1999 while finishing a postdoc at Stanford University. The name ORCA was inspired by a whale watching cruise of Frank Neese at the California coast and stands for itself and the association which comes with it.

ORCA is free for academic use and features a variety of quantum chemical methods including semi-empirical, density functional theory, many-body perturbation, coupled cluster, and multireference methods.

It is mainly developed by the research groups of the Department of Molecular Theory and Spectroscopy at the Max-Planck-Institut für Kohlenforschung and the FACCTs GmbH, which also manages commercial licensing to industry.

ORCA also benefits from a worldwide network of external developers and contributors.

1.2 Installation

1.2.1 Download

ORCA is available for Linux, MacOS, and Windows. Respective installers can be downloaded from the ORCA forum (registration required).

1.2.2 Installation

On Linux and MacOS the most convenient way to install ORCA is by using the command-line installer. You have to download a .run file, e.g. `orca_6_1_0_linux_x86-64_shared_openmpi416.run`. Then make the file executable, that is, open a terminal, enter the directory in which you stored the file and enter:

```
chmod a+x orca_6_1_0_linux_x86-64_shared_openmpi416.run
```

Afterwards you can simple execute the file, like

```
./orca_6_1_0_linux_x86-64_shared_openmpi416.run
```

The installer will install orca in a user directory, as well as set the path to include the orca directory. After *opening a new window* ORCA can be used as indicated.

The installer has a few more options, like extract-only, setting the path interactively, and setting the path by option:

[1] *… so why is it called **ORCA** ?* Frank Neese made the decision to write a quantum chemistry program in the summer of 1999 while finishing a postdoc at Stanford University. While thinking about a name for the program he wanted to write he decided against having yet another "whatever-Mol-something". The name needed to be short and signify something strong yet elegant. During this time in the US Frank went on a whale watching cruise at the California coast—the name "ORCA" stuck. It is often asked whether ORCA is an acronym and over the years, various people made suggestions what acronym this could possibly be. At the end of the day it just isn't an acronym which stands for anything. It stands for itself and the association which comes with it.

```
-i        : Set a different install path interactively
-p <path> : Set a different install path
-x        : Just extract
```

The options have to be given *after* a double dash, e. g.

```
./orca_6_1_0_linux_x86-64_shared_openmpi416.run -- -p /my/home/orca/dir
```

On Window the installer is contained in a zip file `Orca6.1.0.Win64.zip` that must be unzipped before you can start the installation. Than double-click `Orca6.1.0.Win64.exe` and follow the instructions. If you choose a `typical` installation only the serial modules of ORCA are copied. To also install the parallel part of ORCA please use `Custom` or `Full` installation. The ORCA default directory is `C:\Orca_6.1.0` but you can choose a different folder if necessary. The required environment variables will be set accordingly.

Additionally the ORCA AUTOCI executables are provided in 2 different files. `Orca6.1.0.Win64_autoci.zip` contains the modules for serial execution and `Orca6.1.0.Win64_autoci_msmpi10.zip` for the parallel execution. Please extract the zip-files into the ORCA installation directory.

After opening a new command window ORCA can be used as indicated.

> **Note**
>
> The ORCA main modules can be uninstalled via the `Apps & features` function of Windows but the ORCA AUTOCI modules have to be deleted manually.

1.2.3 Path Variables

Only necessary environment variables are added or modified during the ORCA installation.

On Linux:

```
PATH           : updated with the new orca installation directory
```

On Windows:

```
PATH           : modified with the new orca installation directory`
ORCADIR   : created or modified with the new orca installation directory`
XTBEXE    : created or replaced with the xtb module path`
```

> **Note**
>
> Please be aware that the changes of environment variables only effect newly opened command windows.

1.2.4 How do I install the xtb module?

Many feature of ORCA are connected to the xtb module. While there is a native ORCA implementation, the package of the Grimme lab is also integrated into the ORCA workflow, so installing it is advisable. Since ORCA 6 on Linux and Windows the xtb version 7.6.1 is provided with the installation of ORCA. The executable can be found in the orca installation directory (Linux) or in a sub-directory of the orca directory (Windows).

For updates or other xtb versions please download the reqiured xtb version from the Grimme lab's repository Grimme lab's repository and copy the xtb binary into the orca directory and rename it to `otool_xtb` for Linux. On Windows you have to set the XTBEXE variable to point to the xtb executable itself.

```
set XTBEXE=<path>xtb.exe
```

1.3 Graphical User Interfaces (GUI)

Currently, there is no native graphical user interface (GUI) developed by the ORCA developers. However, there are several programs that provide functionalities to create ORCA inputs and read, and visualize ORCA output files. Some useful programs are outlined in Table 1.1 and some applications can be found at the ORCA tutorials website.

Warning

None of the programs mentioned are developed by the ORCA team. We can therefore not guarantee their functionality. Please contact the developers of the respective program in case of problems and feature requests.

Table 1.1: Selected programs that may be used as GUI to ORCA.

Program	Platform	Comment
Avogadro 2	Windows, Linux, MacOS	3D molecule builder, creation of ORCA input, visualization of ORCA outputs.
ChimeraX	Windows, Linux, MacOS	Visualization of ORCA outputs. Very powerful visualization suite. The SEQCROW plugin is very useful. It can be downloaded from the `Tools/More→Tools` menu.
Chemcraft	Windows, Linux	3D molecule builder, creation of ORCA input, visualization of ORCA outputs. For Linux, we recommend to use the Windows version via a virtual machine.

Note

The ORCA enhanced version of Avogadro 1 was not further developed or tested for ORCA 6 and beyond! We recommend to try Avogadro 2 instead.

1.4 Running ORCA

Under Windows the program is called from the command prompt! (Make sure that the PATH variable is set such that the orca executables are visible)

```
orca orca.inp > orca.out
```

Under UNIX based operating systems the following call is convenient (here also: make sure that the PATH variable is set to the directory where the orca executables reside):

```
orca orca.inp >& orca.out &
```

The program writes to *stdout* and *stderr*. Therefore the output must be redirected to the file *orca.out* in this example. *orca.inp* is a free format ASCII file that contains the input description. The program will produce a number of files *orca.x.tmp* and the file *orca.gbw*. The "*.gbw" file contains a binary summary of the calculation. GBW stands for "Geometry-Basis-Wavefunction". Basically this together with the calculation flags is what is stored in this file. You need this file for restarting SCF calculations or starting other calculations with the orbitals from this calculation as input. The "*.tmp" files are temporary files that contain integrals, Fock matrices etc. that are used as intermediates in the calculation. If the program exits normally all of these files are deleted. If it happens to crash you have to remove the files manually (`rm orca*.tmp` under Unix or `del orca*.tmp` under Windows). In case you want to monitor the output file while it is written, you can use the command (under Unix):

```
tail -f orca.out
```

to follow (option -f) the progress of the calculation. Under Windows you have to either open another command shell and use:

```
type orca.out
type orca.out |more
```

or you have to copy the output file to another file and then use any text editor to look at it.

```
copy orca.out temp.out
edit temp.out
```

you cannot use `edit orca.out` because this would result in a sharing violation.

1.4.1 A simple example

A great strength of ORCA is the simple structure of its input file. ORCA input files can be easily generated by any simple text editor. Simply create a file with an `.inp` extension, e.g. `orca.inp`, and add the input information to it. The most basic input file looks like this:

```
! METHOD

*XYZ CHARGE MULTIPLICITY
Atom1   X1  Y1  Y1
Atom2   X2  Y2  Y2
*
```

Here, we define our quantum mechanical method of choice via the so called *simple input line* that starts with an exclamation mark. Afterwards, we define the structure of our molecule, its net charge, and its multiplicity (M = 2S+1). In this example, we chose cartesian coordinates in XYZ format given in Ångström. The coordinates block must be closed by an `*`.

An example, using Hartee-Fock theory and a def2-SVP basis set for the neutral, singlet dihydrogen molecule looks like this:

```
! HF DEF2-SVP

*XYZ 0 1
H   0  0  0
H   0.74  0  0
*
```

More details on the ORCA input file syntax can be found in the section *General Structure of the Input File* and the section *Hello water! Your first ORCA calculation* will guide you through your first steps with ORCA.

1.5 Hello water! Your first ORCA calculation

After *setting up ORCA correctly*, you are ready to perform your first calculation with ORCA! Here, we will exemplarily compute the energy of a water molecule using *Hartree-Fock theory* and the def2-SVP *basis set*. We will use the following `hellowater.inp` ORCA input file:

```
! HF DEF2-SVP

* XYZ 0 1
O   0.0000   0.0000   0.0626
H  -0.7920   0.0000  -0.4973
H   0.7920   0.0000  -0.4973
*
```

Hint

The input usually has the `.inp` extension, but `.txt` is also ok.

You can now run ORCA.

```
orca hellowater.inp > hellowater.out
```

The output is then printed in the file `hellowater.out` in working directory.

```
                                 *****************
                                 * O   R   C   A *
                                 *****************

                                           #,
                                           ###
                                           ####
                                           #####
                                           ######
                                          ########,
                                     ,,################,,,,,
                                ,,#################################,,
                           ,,##########################################,,
                       ,#########################################, ''#####,
                    ,##############################################,,   '####,
                   ,#################################################,,,,####,
                 ,###########''''           ''''################################
                ,#####''   ,,,,##########,,,,          '''####'''          '####
              ,##' ,,,,###########################,,,                        '##
             ' ,,###''''                  '''############,,,
           ,,##''                                '''############,,,,        ,,,,,,###
↪''
      ,#''                                            '''##########################'''
     '                                                          ''''####''''
              ,#######,   #######,   ,#######,      ##
             ,#'     '#,  ##    ##  ,#'     '#,    #''#        ,####,    ,#,
             ##       ##  ##   ,#'  ##           #'  '#       #'       ,# #
             ##       ##  #######   ##           ,######,      #####,      #
             '#,     ,#'  ##    ##  '#,     ,#' ,#      #,     #,   #      #
              '#######'   ##     ##  '#######'  #'      '#     '####' #    #



                 ##########################################################
                 #                        -***-                           #
                 #          Department of theory and spectroscopy         #
                 #                                                        #
                 #                      Frank Neese                       #
                 #                                                        #
                 #     Directorship, Architecture, Infrastructure         #
                 #                    SHARK, DRIVERS                      #
                 #        Core code/Algorithms in most modules            #
                 #                                                        #
                 #        Max Planck Institute fuer Kohlenforschung       #
                 #                Kaiser Wilhelm Platz 1                  #
                 #                 D-45470 Muelheim/Ruhr                  #
                 #                      Germany                           #
                 #                                                        #
                 #                  All rights reserved                   #
                 #                        -***-                           #
                 ##########################################################
```

(continues on next page)

(continued from previous page)

```
[...]
```

You can then open the output file to see the results of your calculation (make sure not to overwrite it while the calculation is running). If the calculation ended normally, a respective message is printed to the end of the output file.

```
                             ****ORCA TERMINATED NORMALLY****
```

The energy of the water molecule can now be found in the output file:

```
-------------------------   --------------------
FINAL SINGLE POINT ENERGY       -75.959340303174
-------------------------   --------------------
```

Congratulations, you successfully ran your first quantum chemical calculation with ORCA!

1.6 General Recommendations

Warning

There are certainly differing opinions among scientists on what method to use when and why. However, we want to give some subjective advise in the context of ORCA based on the developers experience.

ORCA provides a comprehensive and powerful toolkit for quantum chemical calculations. However, the choice of a suitable method for the chemical challenge at hand is often difficult, yet essential to obtain reasonable results. Nevertheless, some general subjective recommendations can be made to navigate through the sheer mass of available method combinations and computational approaches. We further recommend to check for educational review articles that cover the basics of computational chemistry applications.

Educative Reviews on Computational Chemistry

1. Bursch, Markus; Mewes, Jan-Michael; Hansen, Andreas; Grimme, Stefan. Best-Practice DFT Protocols for Basic Molecular Computational Chemistry. *Angew. Chem. Int. Ed.*, **2022**, 61 (42), e202205735. DOI: 10.1002/anie.202205735.

1.6.1 Know Your Methods

Even though the theoretical background of some methods can be complicated, one should know the strengths and limitations of the chosen method. Accordingly, it is recommended to get a fundamental idea of the theoretical basis of any method before using it. Some prominent examples are:

- Typical DFT functionals cannot recover London dispersion effects by theory. Therefore employing a suitable *dispersion correction* is mandatory, especially for large systems.
- Second order Møller-Plesset perturbation theory relies on an energy difference of occupied and unoccupied orbitals in the denominator of its energy expression. Thus, it may break down for systems with small orbital gaps or near-degenerate electronic states (e.g. in many 3d transition metal complexes)
- Single-reference methods like conventional DFT and *single-reference Coupled-Cluster may not be useful for multi-reference cases*. When to go to multireference methods is a more complicated question. Typically, this will be the case if multiplets are desired, pure spin functions for systems with several unpaired electrons, in bond breaking situations or for certain classes of excited states (loosely speaking: whenever there are weakly interacting electrons in the system). However, whenever you decide to do so, please be aware that this require substantial insight into the physics and chemistry of the problem at hand. An uneducated use of *CASSCF* or

MRCI/MRPT method likely yields numbers that are nonsensical and that at tremendous computational cost. Here, there is no substitute for experience (and patience).

Tip

Know the strengths and weaknesses of your method of choice. After all, the accuracy of any approximate method is limited and whether its perfectly suited to compute a desired property can only be answered by experimentation and comparison to experimental results or high-level *ab initio* calculations. In this respect, we recommend to make use of various very useful benchmark studies that provide useful data and method recommendations for various properties!

1.6.2 Cost vs. Accuracy

Time and computer resources are not endless. Accordingly, it is important to know if the method of choice is capable of yielding sufficient accuracy at a given timeframe. Further, the fact that you may be able to compute one or the other number a little more accurate doesn't mean that this helps understanding the physics and chemistry of a target system any better. The danger of getting locked into technicalities and miss the desired insight is real! The most relevant aspects that determine the cost-accuracy-ratio are (a) the method used, (b) the basis set used and (c) the cutoffs and tolerances used. Some example cases are:

- Even high-level electronic structure methods will yield bad or at least unreliable results when used with minimal or small split-valence basis sets like STO-3G or 3-21G. However, they may be useful for very large molecules or in screening and exploratory approaches, if purely qualitative results are required. Basis sets of triple-ζ quality and beyond are recommended to obtain reliable results. Some examples are the Karlsruhe def2-TZVPP and the Dunning cc-pVTZ *basis sets*.
- Using very loose optimization thresholds may speed up the optimization but can prevent your system to reach the real minimum at the given potential energy surface. This is specifically the case for flexible molecules with relatively flat potential energy surfaces.
- Insufficiently tight *numerical integration* grids can speed up the calculation but can also limit the accuracy of otherwise highly accurate method/basis set combinations. Even though, the defaults in ORCA are chosen to give reasonable results in most cases, you should increase the numerical integration grid if you are aiming for high accuracy.
- Some approximations harm more than others. For example, the *RI approximation* combined with sufficiently large auxiliary basis sets
  drastically reduces the computation times without at almost no loss in accuracy. Therefore, we generally recommend the usage of RI techniques in line with ORCA's defaults.

Tip

Be clear about the level of accuracy you want to achieve. If qualitative results are sufficient, it may make sense to reduce the base set size or thresholds. Nevertheless, there are limits beyond which any results are to be considered questionable.

1.6.3 Creating the Right Model

Reproducing properties of a system at experimental conditions, e.g. at elevated temperatures or in solution, is one of the holy grails of quantum chemistry. However, even the best electronic structure method will not help if the model for your system is incomplete or wrong. Some examples are:

- If a molecule is flexible and has a flat potential energy surface, its molecular properties under experimental conditions may not be described by only one conformer. In such cases, multi-structure approaches should be used to describe properties via Boltzmann-weighted conformer ensembles that may be generated with *GOAT*.
- If your experiment is performed in solution, gas-phase simulations may not give reasonably accurate results. Inclusion of solvation effects either by *implicit solvation* or explicit solvent molecules that can be added with the *SOLVATOR* tool may be necessary.
- Creating a smaller model system may reduce the computation times but bears the risk of missing subtile effects that can cause significant differences compared to the experiment.

Tip

Create your model wisely. Always consider inclusion of environmental effects in your calculations and make sure that your structural model can reproduce the desired property.

1.6.4 Matching Method and Property

A method that performs well for one property does not necessarily have to do so for another. This applies in particular to more approximate methods like *DFT*. Therefore, we recommend to consult benchmark studies to find accurate and robust density functional approximations and method combinations for a given property. Some prominent cases are:

- *(meta-)GGA functionals* like PBE or r^2SCAN yield good geometries even when combined with relatively small basis sets. However, for energies at least *hybrid* or even *double-hybrid functionals* are required to reach sufficiently high accuracies.

  Warning

  Some researchers like to adjust the amount of Hartree-Fock exchange according to their needs or what they think is "better" than the standard. This increases the semiempirical character of the calculations and may represent fixes that only work for a given class of compounds and/or properties while worsening the results for others. With this caveat in mind it is one of the things that you are free to try if you like it. However, we do not recommend it since it will deteriorate the comparability of your results with those of other workers the vast majority of which use standard functionals. An alternative to changing the amount of HF exchange could be to simply construct a linear regression for a number of known cases and then use the linear regression.

- Functionals that give mediocre energies like TPSS or TPSSh can be very useful in the context of computing spectroscopic properties like NMR shielding constants or EPR g-Tensors and hyperfine couplings.
- Double-hybrid functionals yield high accuracies for most organic systems but should be avoided for systems with small HOMO-LUMO gaps (e.g. 3d transition metal complexes).

Tip

Check your method of choice. We generally recommend to check for detailed benchmark studies on different properties. If none is available, it may be worth investing the time to benchmark promising methods for your system class and the desired property to compute. If that is not feasible, choosing less empirical higher-level *ab-initio* methods may be the better choice to increase the reliability of the results.

1.6.5 Some Method-specific Recommendations

- DH's The perturbatively corrected functionals (B2PLYP) may also be a very good choice for many problems (at comparable cost to MP2; note that even for large molecules with more than 1000 basis functions the MP2 correction only takes about 10-20% of the time required for the preceding SCF calculation if the RI approximation is invoked. For even larger molecules one has the option of speeding up the MP2 part even further by the DLPNO approximation).
- DLPNO-CCSD(T) Beyond DFT and (SCS-)MP2 there are coupled-cluster methods and their implementation in ORCA is efficient. With the local pair natural orbital methods you can even study molecules of substantial size and with appealing turnaround times.
- **def2-TZVP** is different from the old TZVP. It has been realized that if one invests into an accurate triple-zeta description of the valence region it makes limited sense to only employ a single polarization function. The accuracy is then limited by the polarization set and is not much better than what one gets from SV(P). Hence, def2-TZVP contains a single p-set for hydrogens but is otherwise very similar to the old TZVPP basis set, e.g. it contains 2d1f polarization for main group elements and much more extensive polarization sets for transition metals. The highest polarization function (f for main group) does add substantially to the computational effort. Hence, we often use def2-TZVP without the f polarization function. In order to do that one can use the keyword def2-TZVP(-f). Together with RI or RIJCOSX this is still computationally economic enough for most studies.
- **def2-TZVPP** is a fully consistent triple-zeta basis set that provides excellent accuracy for SCF calculations (HF and DFT) and is still pretty good for correlated calculations. It is a good basis set to provide final single point energies.
- **def2-QZVPP** is a high accuracy basis set for all kinds of calculations. It provides SCF energies near the basis set limit and correlation energies that are also excellent. It is computationally expensive but with RI and RIJCOSX in conjunction with parallelization it can often still be applied for final single-point energy calculations. In conjunction with such large basis sets one should also increase the accuracy of the integration grids in DFT and RIJCOSX — it would be a shame to limit the accuracy of otherwise very accurate calculations by numerical noise due to the grid.
- **Correlation consistent basis sets** provide good correlation energies but poor to very poor SCF energies. For the same size, the ano-pVDZ basis sets are much more accurate but are also computationally more expensive. Except for systematic basis set extrapolation we see little reason to use the cc bases.
- **Pople basis sets** are somewhat old fashioned and also much less consistent across the periodic table than the basis from the Karlsruhe group. Hence, we generally prefer the latter.
- For **scalar relativistic calculations** (X2C,ZORA and DKH) we strongly recommend to use *basis sets that are recontracted for the respective relativistic correction*.
- **Effective core potentials (ECPs)** lead to some savings (but not necessarily spectacular ones) compared to all-electron relativistic calculations. For accurate results, small core *ECPs* should be used. They are generally available for the def2 Karlsruhe type basis sets for elements past krypton. In general we prefer Stuttgart–Dresden ECPs over LANL ones. For the first transition row, the choices are more meager. Here Karlsruhe basis sets do not exist in conjunction with ECPs and you are bound to either SDD or LANL of which we recommend the former. Geometries and energies are usually good from ECPs, but for property calculations we strongly recommend to switch to all electron scalar relativistic calculations using ZORA (magnetic properties) or DKH (electric properties).
- You can take advantage of a built-in basis set (printed using `!PrintBasis` or `orca_exportbasis`) and then modify it by uncontracting primitives, adding steeper functions etc. (fully uncontracted bases are generated via `uncontract` in `%basis`) Alternatively, some basis sets exist that are of at least double-zeta quality in the core region including the DZP and Dunning basis sets. For higher accuracy you may want to consider the `aug-` series of basis sets. See section *Basis Sets* for more about basis set input.
- Likewise, if you are doing calculations on anions in the gas phase it is advisable to include diffuse functions in the basis set. Having these diffuse functions, however, makes things much more difficult as the locality of the basis set is significantly reduced. If these functions are included it is advisable to choose a small value for `Thresh` (10^{-12} or lower). This is automatically done if the smallest eigenvalue of the overlap matrix is below DiffSThresh (which is 1e-6 by default). Also, diffuse functions tend to introduce basis set linear dependency issues, which can be solved by setting `Sthresh` to a larger value than the default 10^{-7} (see

Section *Linear Dependence*). Any value of `Sthresh` beyond 1e-6 has to be used carefully, specially if one is running geometry optimizations, were different basis might be cut off during different geometry steps, or when comparing different conformers since there could be some discontinuity on the final basis set.

- The **integration grids** used in DFT should be viewed together with the basis set. If large basis set calculations are converged to high accuracy it is advisable to also use large DFT integration grids (like `! DEFGRID3`). For "unlimited" accuracy (i.e. benchmark calculations) it is probably best to use product grids (`Grid=0`) with a large value for `IntAcc` (perhaps around 6.0). The default grids have been chosen such that they provide adequate accuracy at the lowest possible computational cost, but for all-electron calculations on heavy elements in conjunction with scalar relativistic Hamiltonians you should examine the grid dependency very carefully and adjust these parameters accordingly to minimize errors. You should be aware that for large molecules the exchange-correlation integration is usually *not* the dominating factor (not even in combination with RI-J).
- Similarly important is the value of `Thresh` that will largely determine the tunaround time for direct SCF calculations. It may be possible to go to values of 10^{-6}–10^{-8} which will result in large speed-ups. However, the error in the final energy may then be 3 orders of magnitude larger than the cutoff or, sometimes, your calculation will fail to converge, due to the limited integral accuracy. In general it will not be possible to converge a direct SCF calculation to better than `Thresh` (the program will also not allow this). For higher accuracy values of maybe 10^{-10}–10^{-12} may be used with larger molecules requiring smaller cutoffs. In cases where the SCF is almost converged but then fails to finally converge (which is very annoying) decreasing `Thresh` and switch to `TRAH` SCF is recommended. In general, `TCut` should be around `0.01` $\times$`Thresh` in order to be on the safe side.
- DFT calculations have many good features and in many cases they produce reliable results. In particular if you study organic molecules it is nevertheless a good idea to check on your DFT results using MP2. MP2 in the form of RI-MP2 is usually affordable and produces reliable results (in particular for weaker interactions where DFT is less accurate). In case of a large mismatch between the MP2 and DFT results the alarm rings — in many such cases MP2 is the better choice, but in others (e.g. for redox processes or transition metal systems) it is not. Remember that SCS-MP2 (RI-SCS-MP2) and double hybrid functionals will usually produce more accurate results than MP2 itself.
- Coupled-cluster calculations become more and more feasible and should be used whenever possible. The DLPNO-CCSD and DLPNO-CCSD(T) calculations are available for single-point calculations and provide accurate results. However, a coupled-cluster study does require careful study of basis set effects because convergence to the basis set limit is very slow. The established basis set extrapolation schemes may be very helpful here. For open-shell molecules and in particular for transition metals one cannot be careful enough with the reference. You have to carefully check that the Hartree-Fock calculation converged to the desired state in order to get coupled-cluster results that are meaningful. Orbital optimized MP2, CASSCF or DFT orbitals may help but we have often encountered convergence difficulties in the coupled-cluster equations with such choices.
- Generally speaking, CEPA is often better than CCSD and approaches the quality of CCSD(T). It is, however, also a little less robust than CC methods because of the less rigorous treatment of the single excitations in relation to electronic relaxation.

1.7 Troubleshooting Problems in ORCA

When starting out with ORCA, trying out features for the first time, or when delving deeper into computational studies, one may run into issues that need troubleshooting. In this section, we summarize typical sources of errors and the error messages encountered to help with troubleshooting common problems.

> **Note**
>
> This section is to be a growing library of examples of common issues. If you are not able to find a solution to your problem here, please post in the ORCA Forum with a general description of your issue and—if possible—the input and/or output files that showcase the issue. If a type of problem comes up often enough, it will be included in this section for future releases!

1.7.1 ORCA Does Not Run...

If ORCA does not seem to be running at all, make sure that the path variables are set correctly (refer to the *Installation* section) and that there are no other programs on your system that interfere with the use of ORCA. One example is if there is another program named ORCA. To check if this is the case, check that `which orca` or `where orca` (or an equivalent thereof) outputs the correct ORCA executable. If it does not, try using the full path to the ORCA executable when trying to run a job. When in doubt, turn to the section on *Running ORCA*.

If ORCA runs for you in serial, but not in parallel, first make sure that you are not running the main ORCA executable with mpirun. The main ORCA driver needs to be run in serial and will call the different modules in parallel if requested in the input file. If you try to run using mpirun, you will typically get the output

```
!!!
```

```
ORCA finished by error termination in <ORCA MODULE>
Calling Command: mpirun -np 8 /path/to/orca/module myjob.moduleinp.tmp myjob
[file orca_tools/qcmsg.cpp, line 394]:
.... aborting the run".
```

At this point, looking at the warnings issued by ORCA at startup can be very helpful. This is located in the output file section titled `WARNINGS`—right after the list of ORCA contributors and before the copy of the input file:

```
============================================================================
                                   WARNINGS
                  Please study these warnings very carefully!
============================================================================
WARNING: Geometry Optimization
  ===> : Switching off AutoStart
         For restart on a previous wavefunction, please use MOREAD
```

This is especially helpful if ORCA found something odd, but decided not to terminate and instead assumed that this was intended by the user.

Other warnings may be issued after the startup warnings. Search the output for other instances of `WARNING` to see what ORCA flagged as possible issues during the calculation. This could be—but is not limited to—non-converging (CAS)SCF, CPSCF, or CC iterations, linear dependencies in the basis set, and memory problems. Typical trademarks of poorly converging geometry and/or SCF runs include, for example:

```
                                   *** Resetting DIIS ***
 WARNING: the maximum gradient error descreased on average only by a factor   0.9
          during the last 20 iterations
                      *** Initiating the TRAH-SCF procedure ***
```

or

```
  0    -4604.304089906417    2.255713e+02                    0.400 (TRAH MAcro)  ␣
↪No
WARNING : negative HOMO - LUMO gap : -60.865582
  0   dE    -8.949363e+01    1.445336e+01    -5.5474e+02     0.400 (TRAH MIcro)
```

Next, check to see if the issue is maybe from outside of ORCA. For example, if you exceed time, disk, or memory limits of the queuing system you use to submit ORCA jobs, the system (or sometimes even an administrator) will terminate your job. In this case, there may be no error message or an error message from your queuing system or system shell. The form of error message you see will depend on your operating system and how the queuing system is set up. The queuing system may also gather the errors in a separate file.

1.7.4 Incorrect Input File Syntax

The following subsections explore common errors with input file syntax. It may be good in general to refer to the sections *General Structure of the Input File* and *Input of Coordinates*.

Unknown Simple Input Keyword

As shown above, the typical error message associated with an unrecognized simple input keyword is

```
            !!!!
```

Unknown Input Block Keyword

Another issue with the input file syntax may be an unrecognized keyword in an input block. The error message for this is typically

```
        Unknown identifier in <BLOCK_NAME> block line <LINE_NUMBER>  :
        Last token : <KEYWORD>
```

If you are sure that the keyword exists, check to make sure that you have closed all input blocks with an `end` statement. Additionally, there may be some subblocks in the input blocks that require an `end` statement.

Required Input is Missing

It is best practice that all ORCA input files specify at least three things: (1) a geometry, (2) a method to run, and (3) a basis set. (1) can be set in the `%coords` block or using the simple input syntax. It may also be read from an external file. (2) and (3) can be set using simple input keywords or their respective input blocks. Refer to the sections *General Structure of the Input File*, *Input of Coordinates*, *Basis Sets*, and the part of the manual for the desired method.

If a method or basis set is not specified, there is often no error message and a default is set. In these cases, it is important to check the output carefully! However, a missing geometry and some auxiliary basis sets do not have defaults and will result in an error. Following are examples of the errors encountered from these issues.

Missing geometry:

```
You must have a [COORDS] ... [END] block in your input
```

Missing auxiliary basis (caught in the main check):

```
ERROR: RI-MP2 needs an AuxC basis but none was defined!
```

Missing auxiliary basis (caught by SHARK):

```
Error (TMOIntegrals): Aux/C requested but not available in SHARK
ORCA finished by error termination in <MODULE>
```

If you are unsure of which auxiliary basis sets to use, a convenient functionality is `!AutoAux`. Note here that these automatically generated auxiliary basis sets may lead to linear dependence issues.

Additionally, if an atom `Xx` is not defined in the provided basis set, it will give the error:

```
                ****************************************************************
                ** There are no main   basis functions on atom number   0 (Xx) **
                ****************************************************************
    The basis set was either not assigned or not available for this element -␣
↪Aborting the run
```

This can be fixed by (1) requesting a different basis set in which that element is defined or (2) using the requested basis set for all other elements and assigning a different basis set just for that element. This can be achieved by using `NewGTO` (see *Assigning Basis Functions to an Element*).

Simply requesting `RI` when the chosen method uses exchange integrals will also lead to ORCA aborting, as it does not know how to handle these integrals. The error message gives the user the options which may be specified:

```
Error: RI is on but the HF exchange must be handled somehow
Use one of the keywords
     RIJDX  : treat the HF exchange exactly (equals RIJONX)
     RIJCOSX: treat the HF exchange by chain-of-spheres
     RIJK   : treat Coulomb and exchange both by RI
```

Depending on the method run and what other keywords are specified, other inputs may be required. In most of these cases, ORCA outputs an error that should be self-explanatory. This error may, for example, look like:

```
        Error: Missing keyword in <BLOCK_NAME> block
        Missing keyword : <KEYWORD>
```

Missing `end` Statement

If an `end` statement is missing to close out an input block, a couple errors may occur. Often, one encounters the `Unrecognized symbol` error, where the next symbol is interpreted as being a part of the previous input block. This error would be of the form:

```
        Unrecognized symbol in <BLOCK_NAME> block line <LINE_NUMBER>  :
        Last token : %
```

One may also get a more obvious error message:

```
        'END' was expected at the end of the <BLOCK_NAME> block line <LINE_NUMBER>
↪  :
        Last token : <PREVIOUS_KEYWORD>
```

Missing Symbols

Just as errors occur with not enough `end` statements, it is also the case with too many `end` statements. For example, some inputs like `%base`, `%moinp`, and `%MaxCore` are global variables and not input blocks. They therefore do not expect `end` statements. In this case, the error may look something like this:

```
ERROR: expect a '$', '!', '%', '*' or '[' in the input
       Line <LINE_NUMBER> of <INPUT_FILE> (END)
```

Here, ORCA was not expecting another keyword and was instead looking for a symbol that would begin a new input block, geometry block, etc.

If instead you provide a keyword outside of an input block, forget the `!` at the beginning of a simple input line or the `*` for the geometry block, or write a comment without a `#`, you will get the same error message as above, just with the first keyword it runs into after the missing symbol in place of `END`.

Trying to read in external files typically requires the file to be provided in quotation marks. If these are missing for the `%moinp` keyword, for example, the error message is:

```
Error: Quotes expected after scanning %%moinp
```

An exception to this is that external coordinate files should not be specified using quotation marks in the simple geometry input section (for example via `*xyzfile`).

Unrecognized Symbols

Another issue is if there are extra symbols than what ORCA is expecting. A common example of this is the simple coordinates input, which begins with an asterisk and ends with an asterisk on a new line if you provide the coordinates in the input file. However, reading coordinates from an external file—using the `*xyzfile` syntax, for example—while also providing a second asterisk (on a new line) gives the error:

```
ERROR: found a coordinate definition line (* ctyp charge mult)
   but could not read ctyp, charge and multiplicity
```

Using `%%` to open an input block instead of `%` or having a stray `%` in the input file results in the error:

```
Error: expected an identifier after '%%' in the input
```

However, using `!!` instead of `!` for the simple input keywords gives an unrecognized keyword error as it interprets the second exclamation mark as its own keyword. Other symbols that ORCA is not expecting will give the error:

```
ERROR: expect a '$', '!', '%', '*' or '[' in the input
```

Cannot Open Files

If you need ORCA to read from another file, you will encounter an error if the file specified does not exist. For example, using `*xyzfile 0 1 geom.xyz` will give the error

```
ORCA_ReadXYZFile::Error
There was a problem reading file: geom.xyz
!!!
```

1.7.5 Out of Memory (OOM)

A typical error message that occurs if `%MaxCore` is not given or if insufficient memory is available for the calculation to proceed is `Please increase MaxCore`. This will typically be the case if ORCA tried to estimate the memory needed and found that `MaxCore` is not sufficient.

> **Note**
>
> `MaxCore` is given in MB and is the amount of maximum memory to allocate **PER PROCESSING CORE**! ORCA may also overshoot `MaxCore` during the calculation and is not necessarily set for all methods. As such, it is generally advised to set `MaxCore` such that `MaxCore * nprocs <= 0.75(Available Memory)`.

For some methods, ORCA tries to estimate the memory needed early on during the calculation. If this estimation is larger than `MaxCore`, but smaller than 2*`MaxCore`, ORCA will issue a warning and proceed. However, if the estimation is larger than 2*`MaxCore`, ORCA will abort. You will then have to increase `MaxCore` in the input file and run the job again.

In some cases, this rough estimation of the memory required is not enough and ORCA realizes during the calculation that it does not have enough memory to proceed. Such an error may look like this:

```
[file orca_tools/qcmem.cpp, line 949, Process 7]:  OUT OF MEMORY ERROR!

--------------------------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code. Per user-direction, the job has been aborted.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun noticed that process rank 0 with PID <PID> on node <NODE> exited on signal␣
↪11 (Segmentation fault).
--------------------------------------------------------------------------
4 total processes killed (some possibly by mpirun during cleanup)

ORCA finished by error termination in SCF
Calling Command: mpirun -np 8  /path/to/orca_scf_mpi myjob.gbw b myjob
[file orca_tools/qcmsg.cpp, line 458]:
  .... aborting the run
```

Note that the `OUT OF MEMORY ERROR` message may not be in the output, depending on the method run.

1.7.6 Disk Full

Some ORCA calculations require a lot of data to be written to disk. This can either be temporary files used during the calculation and deleted afterwards or files that are stored even after the calculation is finished. If the disk becomes full at some point during the calculation, ORCA will abort. This may happen silently (without a clear error message) or with an error message that typically mentions not being able to write a matrix or set a pointer. For example:

```
Error TMatrixContainers::AddMatrix - Failed to add <INDICES> to <TMP_FILE>
```

or

```
Error TMatrixContainers::AddMatrix - Failed to set FilePointer
```

1.7.7 Basis Set Linear Dependence

If some functions in the basis set used are (nearly) linearly dependent, this may lead to numerical issues in the calculation. This is more often the case for basis sets with diffuse functions and auxiliary basis sets generated by `!AutoAux`. ORCA and SHARK try to catch these issues early. For example, you may see

```
WARNING! Potentially linear dependencies in the auxiliary basis
===>>    At the moment, SHARK is not equipped to deal with this
  ... sorry, have to bail out
```

in the output. However, this is sometimes not caught and could lead to numerical instabilities or even ORCA aborting down the line. For example, one may encounter the issue:

```
Diagonalization failed with error code: -5 !
```

One part of the output to check in this case is the diagonalization of the overlap matrix. Very small eigenvalues could point to linear dependence issues. This output may look something like:

```
Diagonalization of the overlap matrix:
Smallest eigenvalue                        ... 3.064e-08
Time for diagonalization                   ... 3.223 sec
Threshold for overlap eigenvalues          ... 1.000e-07
Number of eigenvalues below threshold      ... 3
Smallest eigenvalue above threshold        ... 1.002e-07
```

There are options to help mitigate linear dependence issues, which are explained in more detail in the section on *Linear Dependence*.

1.7.8 Symmetry of Molecule Not as Expected

If you run ORCA with symmetry (via `!UseSym`), be sure to check that the `Auto-detected point group` matches what would be expected. The input geometry may not be perfectly symmetric due to numerical noise. In that case, loosening the threshold for point group identification (for example, with `%sym SymThresh 0.01 end`) should lead to the expected point group and ORCA will use a symmetry-perfected geometry for the calculation. For more information, see the section *ORCA and Symmetry*.

1.7.9 Using Old ORCA Inputs

If you try using an old ORCA input file with a newer version of ORCA, you may find that it does not run. Often when this is the case, there will be an error message issued during the startup of the calculation. This message should help you troubleshoot the issue.

Here, the issue typically arises due to keywords and defaults which may have changed from one version to another. In some cases, a keyword may have even been deprecated. When updating to a new version of ORCA, it is often helpful to refer to the *Detailed change log*, as many of these changes are noted.

If you are trying to read in a GBW file from another version of ORCA, you may find that you run into an error such as:

```
WARNING!!!!!!!
Your GBWFile is either corrupt or from a different ORCA version!
Please be VERY careful with your calculation results!!!

Data_size, sizeof( TOrcaInfo ): 974880, 980832

!!!!!!!
```

(continued from previous page)

```
!!!                          I/O OPERATION FAILED                 !!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

This is especially common when using `!MORead` or when using ORCA utility programs, such as `orca_plot`. In the case of reading in starting orbitals for a subsequent calculation, the additional keyword `!Rescue` may help. Note that with this, fewer checks are performed, so it is important to make sure that the GBW file represents a system that is compatible with the current input file.

For the case of plotting orbitals and certain densities with `orca_plot`, you can either use the same ORCA version as the one that produced the GBW file or you can run a calculation with the newer ORCA version by using `!Rescue MORead NoIter` in the input file (and reading in the GBW file via the `%moinp` keyword). This calculation should not take long and the new GBW file produced should be readable by `orca_plot`. Note that in this case, all other settings should be specified as they were for the calculation that produced the older GBW file.

If you are unsure about an input, please first consult the relevant parts of this ORCA Manual. It is provided by the ORCA developers and should contain all information about the functionality implemented in this version of ORCA. It is also important to reference the ORCA Manual for the version of ORCA that you are using. If you are viewing this manual online and are not using the most recent release version of ORCA, please update your ORCA version or find the ORCA Manual for the version you are using. If you are still unable to get an input file to work, the ORCA Forum is a great place to turn to for getting answers to your questions!

1.7.10 Reproducing Results of Other Program Packages or Older ORCA Versions

Reproducibility of results is fundamental in science, so it is good practice to compare results when switching from one program to another or from one ORCA version to another. If you are trying to do this and observe deviations that are larger than expected, here are some hints and tips.

Newer versions of ORCA might include changes to the default settings, which may affect the numerics. For example, ORCA 6 has improvements to the numerical integration grids. In most cases, these changes should lead to better results and are most likely the reason for differences between the latest ORCA release and previous versions. It is usually possible to manually set these values to their old defaults for comparison purposes. For such an inspection, compare the information in the output to see which values have changed. Reference the relevant sections of this manual to determine how to change these values.

In trying to reproduce results from another program in ORCA (or vice versa), it is important to check that the methods being used are in fact equivalent and that both programs compute the value(s) in question in the same manner. This may mean using some settings that you would not otherwise use so that you can make sure that the two should be comparable. For example, ORCA uses pure d- and f-functions (i.e. 5 d-functions and 7 f-functions instead of the Cartesian 6 d-functions and 10 f-functions) for all basis sets, so this should be taken into account when comparing to programs that do not.

If you still run into issues with reproducibility, open a post in the ORCA Forum and give enough details on what is being compared so that other users and developers can be of assistance. It is often helpful to include the input(s) and/or output(s).

1.7.11 Orbitals Not Found

If you encounter this error:

```
No orbitals were found in the gbw file
```

it is most likely that you are trying to use `!MORead` to read in a GBW file with the same basename as the current input file. This is done automatically with ORCA's AutoStart feature (on by default), so you should be able to simply remove the `!MORead` and `%moinp` parts of the input file. If you want to save the previous orbitals (for later access) and read them in, copy the GBW file to something with a different basename and then read that file in as usual.

1.7.12 Trouble Restarting Input File

Another AutoStart issue is when you make a significant change to your input file and try to rerun it without deleting the old ORCA outputs. This often happens if there is a significant mismatch in the geometry—like when atoms have been reordered or removed. In this case, the error encountered is:

```
Error: Input geometry does not match current geometry
```

Typically, the basis set can be changed without issue, though, as ORCA will perform a projection of the old orbitals onto the new basis set if needed.

If there is an issue with restarting a file, you can always try to run the job from scratch by adding `!NoAutoStart` to the input file or by deleting all other files with the same basename as the input file before rerunning.

1.7.13 SCF Convergence Problems

A typical problem that occurs for systems with complex electronic structures like transition metal complexes or systems with many orbitals in a narrow energy range (such as metal clusters) is a poorly-converging SCF. While ORCA offers keywords like `!SlowConv`, this might not be the best option. Specifically, `!SlowConv` may converge to a local minimum solution that is closer to that of the initial guess. We have compiled a list of tips and tricks in the section *Converging SCF Calculations*.

Example: Converging DFT for Open-Shell Transition Metals

Since DFT is frequently applied to open-shell transition metals, we show here a (more or less trivial) example of a Cu(II) complex treated with DFT:

```
*xyz -2 2
  Cu  0     0     0
  Cl  2.25  0     0
  Cl -2.25  0     0
  Cl  0     2.25  0
  Cl  0    -2.25  0
*

%compound
NewStep
  !BP86 SV SlowConv
StepEnd

ReadMOs(1);
NewStep
  ! B3LYP NoRI TZVP TightSCF
  %scf GuessMode CMatrix end
StepEnd
EndRun
```

Although it would not have been necessary for this particular example, this shows a possible approach to converging such calculations. We first perform a less accurate (but quicker) job using the RI approximation, a GGA functional, and a small basis set without polarization functions. Note that a larger damping factor has been used in order to guide the calculation (`!SlowConv`).

The second job takes the orbitals of the first as input and performs a more accurate hybrid DFT calculation. A subtle point in this calculation on a dianion in the gas phase is the input specification `GuessMode CMatrix`, which requests the use of the corresponding orbital projection. This matches the orbitals of the smaller basis set calculation to those of the larger basis set. This is always required when the orbital energies of the small basis set calculation are positive, as will be the case for anions.

1.7.14 Troublesome Geometry Optimizations

While some issues during optimization jobs give errors that have been covered above, there are problems that may arise which do not produce error messages or terminate ORCA. In some cases, you may get a normal termination, but the results you obtain are not valid. One must often use chemical intuition to determine if the geometry optimization yields reasonable structures and frequencies.

Imaginary Frequencies after Optimization

If a geometry optimization terminates normally, the algorithm has decided that it is sufficiently close to a stationary point on the potential energy surface. To test whether this stationary point is a minimum, maximum, or transition state or to check how close the structure is to the stationary point, a frequency calculation must be run. This might yield imaginary frequencies (which arise due to negative force constants as the frequencies are proportional to the square root of them), which indicates that we are not at a minimum.

```
-----------------------
VIBRATIONAL FREQUENCIES
-----------------------

Scaling factor for frequencies =  1.000000000  (already applied!)

   0:         0.00 cm**-1
   1:         0.00 cm**-1
   2:         0.00 cm**-1
   3:         0.00 cm**-1
   4:         0.00 cm**-1
   5:         0.00 cm**-1
   6:        -5.66 cm**-1 ***imaginary mode***
   7:        14.88 cm**-1
   8:        21.54 cm**-1
```

There are several approaches to deal with this issue. Typically, using tighter optimization criteria, tighter DFT grids, and/or tighter SCF convergence criteria helps, especially for small imaginary frequencies. Another approach can be to identify the normal modes associated with imaginary frequencies, distort the molecule along these vectors, and re-optimize the structure. Note that the compound script *Iterative optimization* does this exact process in an automated fashion.

Getting Unwanted Local Minima

If the optimized geometry you are getting is not the one you were expecting, you may be running into a nearby local minimum on the potential energy surface. This is often encountered when optimizing a molecule that has many rotamers.

First, check that the input geometry you have provided is reasonably close to the conformation you are aiming for. It may help to preoptimize the structure with an inexpensive method. If aiming for the global minimum, one may make use of *GOAT* to preoptimize the structure to a reasonable guess for the global minimum.

If there is symmetry breaking during the optimization, but you would like to enforce the original point group, look at the options in *Geometry optimizations using symmetry*.

Issues with Optimization Convergence

There are a few options you have when the calculation simply does not meet the optimization convergence criteria by the set number of maximum iterations. Inspect the last single point geometry along the optimization trajectory. If it seems intuitively close to a stationary point, try to rerun the optimization from this geometry. This is equivalent to rerunning the original input file, but increasing the number of iterations in the optimization.

Another approach is to first optimize with a less expensive method and/or smaller basis set, then use that geometry for the optimization at the level of theory wanted. The optimization can also first be done with looser convergence thresholds before tightening them as needed.

If none of these tips help, one may identify parts of the structure that are giving issues by analyzing the trajectory of the optimization. The optimization may have gotten stuck at the end in a region where the potential energy surface is nearly flat or has many shallow minima—such as for structures with many rotamers or for systems with multiple molecules. In that case, some reasonable geometry constraints may be needed. It may also be wise to change the trust radius or the maximum step in internal coordinates. In some cases, cartesian optimization via `!COPT` may even be needed.

For more details, see the section *Geometry Optimizations*, especially for the *Geometry Optimization Keywords*.

1.7.15 "Strange" Results for Energies and Properties

If results for energies or properties seem off, first make sure that the geometry, spin multiplicity, and charge used are reasonable for the system at hand. Make sure that the input is a good approximation of the real system you are trying to model. To this end, consider if a solvent model is appropriate. This is especially important for charged systems. For mean-field methods, also consider using a dispersion correction if not already included. If heavy elements are included in the system, scalar relativistic methods may be appropriate, and possibly picture change and finite nucleus corrections.

Next, make sure that the SCF has converged well. For properties and post-HF energies and properties, the SCF should be converged tightly enough that the subsequent calculations do not suffer from the numerical noise of the underlying SCF results. A loose convergence threshold will ultimately affect the properties that follow from the results of the SCF.

For unrestricted methods, it is advisable to look at the calculated spin contamination in the output. If this is significant, results for the given system with that method may be unreliable. One should be careful in the analysis of the output file in this case. In some cases, the underlying representation of the system is flawed or incomplete, but in other cases, you may consider turning to methods that give pure spin state solutions, such as ROHF, ROKS, or multireference methods.

In general when using single reference methods, one should question if the system may exhibit multireference character. In such a regime, results from single reference methods may not be reliable. To help determine this, there are a couple diagnositics that can be computed. When running a coupled cluster calculation, the T_1 diagnostic is commonly used to estimate the importance of static correlation in the description of the system. The computed value is printed when running a coupled cluster calculation and if greater than 0.02, the system **may** have multireference character. More reliable is the newer S-diagnostic, which can be requested by `%mdci DoSDiagnostic true end`. See the *Coupled Cluster S-Diagnostic* section for more information.

When running a multireference calculation, it is important to make sure that the final active space is reasonable and includes all important orbitals that contribute to the static correlation. This may require changes to `nel` and `norb`. Good starting orbitals are important here (see the section on *Choice of Inital Guess*). Starting orbitals may also be generated from other calculations—for example using MP2 natural orbitals or QROs from UKS—which often represent a more physically relevant starting guess. *AVAS* is helpful for building the active space from these starting guesses, provided that you can intuitively determine the AOs that contribute to the MOs you want to include in the active space. Note that although an active space may seem reasonable, there may be convergence issues in CASSCF or in the post-CASSCF calculations when some of the active orbitals are very highly occupied/unoccupied. Finally, one should also consider if a state-specific or state-averaged solution is most appropriate for the system. Keywords which may need to be changed here are `nroots`, `mult`, `weights`, and `irrep`.

Finally, it should be checked that the methods used are appropriate for the system at hand. For example, MP2—and CCSD(T) to a lesser extent—has issues with orbital degeneracies. For properties, linear response and sum-over-states

results are only reliable for systems without low-lying excited states. If the system is outside of the regime where the requested method is reliable, another method should be considered.

If everything seems reasonable, but there still seems to be an issue, post the case in the ORCA Forum. It could be that there is a technical reason for the values to be how they are or there may be a bug that should be fixed.

CHAPTER

TWO

ESSENTIAL CALCULATION ELEMENTS

2.1 General Structure of the Input File

In general, the input file is a free format ASCII file and can contain:

- one or more *"simple" keyword lines* that start with a "`!`" character,
- one or more *input blocks* enclosed between an "`%`" sign and "`end`" that provide finer control over specific aspects of the calculation,
- the *specification of the coordinates*, total charge, and spin multiplicity for the system, either with a `%coords` block, or more usually enclosed within two "`*`" symbols.

Here is an example of a simple input file that contains all three input elements:

```
! HF def2-TZVP

%scf
   convergence tight
end

* xyz 0 1
C  0.0  0.0  0.0
O  0.0  0.0  1.13
*
```

Comments in the file start by a "#". For example:

```
# This is a comment. Continues until the end of the line
```

Comments can also be closed by a second "#", as the example below where `TolE` and `TolMaxP` are two variables that can be user specified:

```
TolE=1e-5;    #Energy conv.#  TolMaxP=1e-6; #Density conv.#
```

The input may contain several blocks, which consist of logically related data that can be user controlled. The program tries to choose sensible default values for all of these variables. However, it is impossible to give defaults that are equally sensible for all systems. In general the defaults are slightly on the conservative side and more aggressive cutoffs etc. can be chosen by the user and may help to speed things up for actual systems or give higher accuracy if desired.

Note

- The ORCA input is **NOT** case sensitive. UPPER CASE, lower case, or aNy cOMmBINAtiON are allowed. An exception is file names (e.g. for `%MOInp` or `*XYZName`), which are case-sensitive on Unix-like OSs.
- In general, the order of the simple keywords and input blocks is not important – see *Input Priority and Processing Order* for the finer details.

2.1.1 Input Blocks

Input blocks start with "%", followed by the block name and end with "end". For example:

```
%method
  method HF
END
```

A list of available input blocks is given in Section 2.1.6. No blocks *need* to be present in an input file but they *can* be present if detailed control over the behavior of the program is desired. Otherwise, most jobs can be defined via the simple keywords described in Section 2.1.7. Variable assignments within blocks have the following general structure:

```
VariableName Value
# or with an optional "=" sign
OtherVariableName = OtherValue
```

Values can be either numeric, quote-delimited strings (see below), or predefined aliases (such as HF in the example above), which are internally converted to some numeric representation.

Some variables are actually arrays. In this case several possible assignments are useful:

```
Array[1]  Value1
Array[1]  Value1,Value2,Value3
Array  Value1,Value2
```

Note

Arrays always start with index 0 in ORCA (this is because ORCA is a C++ program). The first line in the example gives the value "Value1" to Array[1], which is the *second* member of this array. The second line assigns Value1 to Array[1], Value2 to Array[2] and Value3 to Array[3]. The third line assigns Value1 to Array[0] and Value2 to Array[1].

Strings (such as filenames) must be enclosed in quotes. For example:

```
%scf
  MOInp  "Myfile.gbw"
end
```

Note that file names on Unix-like systems are case-sensitive (i.e., MYFILE.GBW and MyFile.gbw are different files). Under Windows the file names are not case sensitive.

Some input block keywords either open a nested sub-block, which must be closed with an additional end, or have a specific syntax, different from the simple variable assignment described above. For example:

```
%scf
  Guess PModel  # variable assignment
  SOSCF  # nested sub-block
    start 0.002  # variable assignment
  end  # closes the SOSCF sub-block
end
%mdci
  # special syntax
  MP2FragInter {1 1} {2 2}
end
%basis
  NewGTO  # nested sub-block
    # special syntax inside
    H "def2-SVP"
    S 1
    1 0.05 1.0
```

(continues on next page)

(continued from previous page)

```
  end  # closes the NewGTO sub-block
end
```

Finally, there are input "blocks" which only set a single variable and are not closed with "`end`". These are listed in Table 2.2. For example:

```
%MOInp "MyFile.gbw"
%maxcore 3000
```

2.1.2 Input Priority and Processing Order

In more complicated calculations, the input can get quite involved. Therefore it is worth knowing how it is internally processed by the program:

- First, all the simple input lines (starting with "`!`") are collected into a single string.
- The program looks for all known keywords in a predefined order, regardless of the order in the input file.
- An exception are basis sets: if two different orbital basis sets (e.g. `! def2-SVP def2-TZVP`) are given, the latter takes priority. The same applies to auxiliary basis sets of the same type (e.g. `! def2/J SARC/J`).
- Some simple input keywords set multiple internal variables. Therefore, it is possible for one keyword to overwrite an option, set by another keyword. We have tried to resolve most such cases in a reasonable way (e.g. the more "specific" keyword should take precedence over a more "general" one) but it is difficult to foresee every combination of options.
- Next, the block input is parsed in the order it is given in the input file.
- Most block input keywords control a single variable (although there are exceptions). If a keyword is duplicated, the latter value is used.
- In principle, the same block may exist multiple times with different variables set within. However, some blocks "reset" certain internal data whenever they are opened, or have certain mandatory contents, which must be present in each block instance (and override previous instances). Therefore, it is not recommended to have multiple instances of the same block.

Consider the following (bad) example:

```
! def2-TZVP UKS
%method
  functional BP86
  correlation C_LYP
  SpecialGridAtoms[1]  26, 27
  SpecialGridIntacc 8,  8,  8
  SpecialGridAtoms 28, 29
end
! PBE def2-SVP RKS
```

Using the rules above, one can figure out why it is equivalent to this one:

```
! UKS BLYP def2-SVP
%method
  SpecialGridAtoms 28, 29, 27
  SpecialGridIntacc 8,  8,  8
end
```

2.1.3 Global Memory Use

Some ORCA modules (in particular those that perform some kind of wavefunction based correlation calculations) require large scratch arrays. Each module has an independent variable to control the size of these dominant scratch arrays. However, since these modules are never running simultaneously, we provide a global variable `MaxCore` that assigns a certain amount of scratch memory to all of these modules. Thus:

```
%MaxCore 2000
```

sets 2000 MB as the limit for these scratch arrays. **This limit applies per processing core**. Do not be surprised if the program takes more than that – this size only refers to the dominant work areas. Thus, you are well advised to provide a number that is no more than 75-80% of your physical memory. Some memory-hungry operations will take longer if given less than the required memory, while others will abort completely if `MaxCore` is insufficient. The default value is 4GB, which is plenty for most standard DFT calculations. For coupled clusters and the like, at least 8GB are recommended.

2.1.4 Changing the Default BaseName

ORCA generates a number of output files, as well as many temporary files, which are removed at the end of a successful run. To prevent filename clashes, all generated files start with the same prefix or *BaseName*. This is usually inferred from the name of the input file by removing the extension, i.e. running ORCA with `MyJob.inp` will create `MyJob.gbw`, `MyJob.properties.txt`, etc. It is also possible to set the BaseName explicitly using the `%base` variable. In the following example, the names of all generated files will start with `job1`, regardless of the name of the input file:

```
%base "job1"
```

2.1.5 Jobs with Multiple Steps

Warning

The `$new_job` feature is a deprecated function. Using `$new_job` might result in erratic results and strange behavior of succeeding calculations. Please use the *compound* feature of ORCA for tasks like this – it is safer and by far more powerful!

ORCA supports input files with multiple jobs. This feature is designed to simplify series of closely related calculations on the same molecule or calculations on different molecules. The objectives for implementing this feature include:

- Calculate of a molecular property using different theoretical methods and/or basis sets for one molecule.
- Calculations on a series of molecules with identical settings.
- Geometry optimization followed by more accurate single points and perhaps property calculations.
- Crude calculations to provide good starting orbitals that may then be used for subsequent calculations with larger basis sets.

For example consider the following job that in the first step computes the g-tensor of BO at the LDA level, and in the second step using the BP86 functional.

```
# ----------------------------------------------------
! LSD DEF2-SVP TightSCF KeepInts
# ----------------------------------------------------
%eprnmr gtensor 1 end

* int 0 2
   B  0  0  0   0      0  0
```

(continues on next page)

(continued from previous page)

```
   O  1  0  0   1.2049 0  0
*

# ***************************************************
# ****** This starts the input for the next job   *
# ***************************************************
$new_job
# ---------------------------------------------------
! BP86 DEF2-SVP SmallPrint ReadInts NoKeepInts
# ---------------------------------------------------
%eprnmr gtensor 1 end

* int 0 2
   B  0  0  0   0      0  0
   O  1  0  0   1.2049 0  0
*
```

What happens if you use the `$new_job` feature is that all calculation flags for the actual job are transferred from the previous job and that only the changes in the settings must be input by the user. Thus if you turn on some flags for one calculation that you do not want for the next, you have to turn them off again yourself (for example the use of the RI approximation)! In addition, the default is that the new job takes the orbitals from the old job as input. If you do not want this you have to overwrite this default by specifying your desired guess explicitly.

2.1.6 List of Input Blocks

Table 2.1 lists the known input block names, along with any accepted synonyms/aliases. The keywords defined in each block are listed in the respective section of the manual and references to these lists are also given in the table. The list of "blocks" which are not closed with `end` is given in Table 2.2.

Table 2.1: Input block keywords. Synonyms are given in parentheses.

Block	Description (Keyword Reference)
`autoci`	Autogenerated single- and multi-reference correlation methods (Section 3.11.9)
`basis`	Basis sets (Table 2.41)
`casresp`	CASSCF static linear response (Section 5.26.6)
`casscf`	CASSCF/NEVPT2 and DMRG calculations (Section 3.13.4)
`chelpg`	CHELPG charges (Section 5.1.8)
`cim`	Cluster-in-molecules calculations (Section 3.10.9)
`cis (tddft)`	CIS and TD-DFT calculations (Section 5.6.19)
`compound`	Compound jobs (Section 8.2)
`conical`	Optimization of conical intersections (Section 4.9)
`coords`	Input of atomic coordinates (Section 2.2)
`cosmors`	OpenCOSMO-RS options (Section 2.13.6)
`cpcm`	Conductor-like Polarizable Continuum Model (Section 2.13.8)
`docker`	Host-guest docking algorithm (Table 4.12)
`eda`	(Table 5.28)
`elprop`	Electric properties (Section 5.20.2)
`eprnmr`	EPR and NMR properties (Section 5.21.9; g-tensor: Table 5.13, HFC: Table 5.14, ZFS: Table 5.16, Mössbauer: Table 5.18)
`esd`	Excited state dynamics (Section 5.5.12)
`frag`	Automatic fragmentation procedure (Table 2.62)
`freq`	Vibrational frequencies (Section 4.7.4)

continues on next page

Table 2.1 – continued from previous page

Block	Description (Keyword Reference)
`geom`	Geometry optimization (Table 4.3; Scan: Table 4.4, TS: Table 4.5)
`goat`	Global optimization algorithm (Table 4.9)
`ice (iceci,cipsi)`	Iterative configuration expansion CI calculations (Section 3.14.11)
`irc`	Intrinsic reaction coordinate calculations (Section 4.4.2)
`lft`	Ligand field theory utility `orca_lft` (Section 9.2.15)
`loc`	Localization of orbitals (Section 9.2.5)
`mcrpa`	CASSCF linear response (Section 5.8.6)
`md`	Molecular dynamics (Table 7.1)
`mdci`	Single reference correlation methods (Section 3.10.14; RHF EOM-CC: Table 5.11, UHF EOM-CC: Table 5.12, STEOM-CC: Section 5.10.1, MR-EOM-CC: Section 3.20.4, LED: Section 5.38.7, ADLD/ADEX: Section 5.38.8, HFLD: Table 5.37)
`mecp`	Minimum energy crossing points optimization (Section 4.10.1)
`method`	Choice of computation method and various options • Run types and method classes: Table 2.5 • RI/COSX: Table 2.45 • Grids: Section 2.10.5 • CP-SCF: Section 2.22 • Frozen core: Section 2.21 • Population analysis: Table 5.3 • DFT: Table 3.3 • DFT/hybrid: Table 3.5 • DFT/range-separated: Table 3.7 • DFT/double-hybrid: Table 3.10 • DFT/LibXC: Section 3.3.6 • DFT/dispersion: Table 3.14 • DFT/NLC: Table 3.20 • gCP: Table 2.52 • NDO: Table 3.23 • Native xTB: Table 3.28 • FMM: Section 6.5.4
`mm`	Molecular mechanics force-fields (Section 3.23.5)
`mp2`	MP2 calculations (Section 3.9.1; DLPNO: Section 3.9.10, DLPNO/gradient: Section 3.9.11, DLPNO/response: Section 3.9.11, DLPNO/multi-level: Section 3.9.11, OO-RI-MP2: Section 3.9.14, regularized: Section 3.9.15)
`mrcc`	Multi-reference CC calculations (Section 3.12)
`mrci`	Multi-reference CI calculations (Section 3.19.15)
`mtr`	Normal mode trajectory/scan (Section 5.6.17)
`nbo`	NBO analysis (Section 5.2)
`ndoparas`	Parameters for NDO-based semi-empirical methods (Table 3.21)
`neb`	NEB calculations (Section 4.6.14)
`numgrad`	Numerical gradients (Section 2.23.2)
`output`	Control of output (Table 2.6)
`pal`	Parallel jobs (Section 2.5)
`paras`	Input of geometric parameters, equivalent to `paras` or `pardef` in `%coords` (Section 2.2.4)

continues on next page

Table 2.1 – continued from previous page

Block	Description (Keyword Reference)
plots	Plot generation (Section 9.1)
qmmm	Multiscale (QM/MM) calculations (Section 6.1)
rel	Relativistic options (Table 2.55; SOC: Section 5.28)
rocis	Restricted-open-shell CIS (Section 5.7.7)
rr	Resonance Raman and absorption/fluorescence band-shape calculations via orca_asa (Section 5.16.4)
scf	SCF procedure settings (Section 2.6; TRAH-SCF: Section 2.6.7, AVAS: Section 3.13.7, ROHF: Section 3.1.1, native xTB: Table 3.29, ΔSCF: Table 5.27, initial guess: Section 2.20, rotate MOs: Section 2.20.7)
shark	SHARK integral package (Section 2.15.6)
solvator	Explicit solvation algorithm (Table 4.10)
symmetry (sym)	Spatial symmetry recognition (Table 2.67)
vpt2	Vibrational perturbation theory (Section 5.19)
xtb	Options for the xtb program interface (Table 3.26)

Table 2.2: Input keywords, which are prefixed with % but have no closing end.

Keyword	Value	Description (Reference)
base	"<BaseName>"	Base name for the files created by the job (Section 2.1.4)
cclib	"<FileName>"	File with one-particle coupling coefficients for ICE-CI (Section 3.14.12)
id	"<string>"	**Deprecated!** Identifier for the job used to summarize computed energies (Section 2.11
ljcoefficients	"<FileName>"	File with Lennard–Jones coefficients for PHVA (Section 4.3.5)
maxcore	4096	Maximum heap memory to use in MB, default: 4GB (Section 2.1.3)
moinp	"<FileName>"	GBW file from which to read guess MOs for the MORead guess (Section 2.20.5)
pointcharges	"<FileName>"	File to read external point charges from (Section 2.2.6)

2.1.7 Simple Keyword Lines

It is possible to give a line of keywords that assign certain variables that normally belong to different input blocks. The syntax for this "simple input" is line-oriented. A keyword line starts with the "!" sign and can contain any number of space-separated keywords. The input file can contain any number of keyword lines and they do not need to be at the beginning (although that is common practice) – see also *Input Priority and Processing Order*.

```
! Keyword1 Keyword2
! Keyword3
```

Most simple input keywords are documented in the relevant section of the manual. Table 2.3 provides references to these sections, grouped by topic. Since simple keywords are usually related to variables within one or more input blocks, it is also instructive to consult the documentation for the respective block – see Table 2.1.

Table 2.3: References to the documentation of simple input keywords. Related input blocks are also listed.

Keyword Group	Reference	Input block
Run types and method classes	Table 2.4	method
Basis sets	Table 2.40	basis

continues on next page

Table 2.3 – continued from previous page

Keyword Group	Reference	Input block
DFT functionals	• LDA: Table 3.1 • GGA: Table 3.2 • global hybrid: Table 3.4 • range-separated hybrid: Table 3.6 • global double-hybrid: Table 3.8 • range-separated double hybrid: Table 3.9 • LibXC: Table 3.12 • Non-local correlation: Table 3.19	`method`
Dispersion corrections	Table 3.13	`method`
Composite (3c) methods	Table 3.32	
NDO-based semi-empirical methods	Table 3.27	`method`
Native xTB-based methods	Table 3.22	`method`
Second order Møller–Plesset perturbation theory	Table 3.34	`mp2`
Single-reference correlated methods via MDCI	Table 3.43	`mdci`
Correlated methods using automatic code generation (AUTOCI)	Section 3.11.2	`autoci`
CASSCF	Table 3.47	`casscf`
Multireference correlated methods via MRCI	Table 3.52	`mrci`
Excited states via correlated wavefunction-based methods	• EOM-CC: Table 5.10 • STEOM-CC: Section 5.10 • others: Section 5.4	`mdci`
Initial guess	Table 2.68	`scf`
SCF procedure	Table 2.11	`scf`
Integral approximations (RI/COSX)	Table 2.44	`method`
Relativistic methods	Table 2.54	`rel`
Implicit solvation (CPCM)	Table 2.56	`cpcm`
Spin-orbit coupling operator	Table 5.19	`rel`
Numerical integration grids	Table 2.48	`method`
Integral storage and handling	Table 2.58	`scf`
Population analysis	Table 5.1	`output`
Output control	Table 2.8	`output`
Nudged elastic band method	Section 4.6.14	`neb`

2.2 Input of Coordinates

Coordinates can be either specified directly in the input file or read from an external file, and they can be in either Cartesian ("xyz") or internal coordinate format ("Z-matrix").

2.2.1 Reading coordinates from the input file

The easiest way to specify coordinates in the input file is by including a block like the following, enclosed by star symbols:

```
* CType Charge Multiplicity
...
coordinate specifications
...
*
```

Here `CType` can be one of `xyz`, `int` (or `internal`), or `gzmt`, which correspond to Cartesian coordinates, internal coordinates, and internal coordinates in Gaussian Z-matrix format.

The input of Cartesian coordinates in the "`xyz`" option is straightforward. Each line consists of the label for a given atom type and three numbers that specify the coordinates of the atom. The units can be either Ångström (default) or Bohr. This can be specified via the simple keywords `!Angs` or `!Bohrs`, respectively, or via the variable `Units` in the `%coords` block described below.

```
* xyz Charge Multiplicity
Atom1   x1  y1  z1
Atom2   x2  y2  z2
 ...
*
```

For example for CO^+ in a $S = 1/2$ state (multiplicity = $2 \times 1/2 + 1 = 2$)

```
* xyz 1 2
C  0.0  0.0  0.0
O  0.0  0.0  1.1105
*
```

Internal coordinates are specified in the form of the familiar "Z-matrix". A Z-matrix basically contains information about molecular connectivity, bond lengths, bond angles and dihedral angles. The program then constructs Cartesian coordinates from this information. Both sets of coordinates are printed in the output such that conversion between formats is facilitated. The input in that case looks like:

```
* int Charge Multiplicity
Atom1  0  0  0    0.0   0.0   0.0
Atom2  1  0  0    R1    0.0   0.0
Atom3  1  2  0    R2    A1    0.0
Atom4  1  2  3    R3    A2    D1
. . .
AtomN  NA NB NC   RN    AN    DN
*
```

The rules for connectivity in the "`internal`" mode are as follows:

- `NA`: The atom that the actual atom has a distance (`RN`) with.
- `NB`: The actual atom has an angle (`AN`) with atoms `NA` and `NB`.
- `NC`: The actual atom has a dihedral angle (`DN`) with atoms `NA`, `NB` and `NC`. This is the angle between the actual atom and atom `NC` when looking down the `NA-NB` axis.
- Note that - contrary to other parts in ORCA - atoms are counted starting from 1.

Angles are always given in degrees! The rules are compatible with those used in the well known MOPAC and ADF programs.

Finally, `gzmt` specifies internal coordinates in the format used by the Gaussian program. This resembles the following:

```
* gzmt 0 1
    C
    O   1   4.454280
    Si  2   1.612138    1   56.446186
    O   3   1.652560    2   114.631525  1   -73.696925
    C   4   1.367361    3   123.895399  2   -110.635060
    ...
*
```

An alternative way to specify coordinates in the input file is through the use of the `%coords` block, which is organized as follows:

```
%coords
 CTyp   xyz      # the type of coordinates = xyz or internal
 Charge 0        # the total charge of the molecule
 Mult   2        # the multiplicity = 2S+1
 Units  Angs     # the unit of length = angs or bohrs

 # the subblock coords is for the actual coordinates
 # for CTyp=xyz
  coords
     Atom1   x1  y1  z1
     Atom2   x2  y2  z2
  end
 # for CTyp=internal
  coords
     Atom1  0  0  0    0.0   0.0   0.0
     Atom2  1  0  0    R1    0.0   0.0
     Atom3  1  2  0    R2    A1    0.0
     Atom4  1  2  3    R3    A2    D1
      . . .
     AtomN  NA NB NC   RN    AN    DN
  end
end
```

Important

Since ORCA is a C++ based program its internal counting starts from zero. Therefore all electrons, atoms, frequencies, orbitals, excitation energies etc. are counted from zero. User-based counting such as the numeration of fragments is counted from one.

2.2.2 Reading coordinates from external files

It is also possible to read the coordinates from external files. The most common format is a `.xyz` file, which can in principle contain more than one structure (see section *Multiple XYZ File Scans* for this multiple XYZ feature):

```
* xyzfile Charge Multiplicity Filename
```

For example:

```
* xyzfile 1 2 mycoords.xyz
```

A lot of graphical tools like Gabedit, molden or Jmol can write Gaussian Z-Matrices (`.gzmt`). ORCA can also read them from an external file with the following

```
* gzmtfile 1 2 mycoords.gzmt
```

Note that if multiple jobs are specified in the same input file then new jobs can read the coordinates from previous jobs. If no filename is given as fourth argument then the name of the actual job is automatically used.

```
... specification for the first job

$new_job
! keywords
* xyzfile 1 2
```

In this way, optimization and single point jobs can be very conveniently combined in a single, simple input file. Examples are provided in the following sections.

2.2.3 Special definitions

- **Dummy atoms** are defined in exactly the same way as any other atom, by using "DA", "X", or "Xx" as the atomic symbol.
- **Ghost atoms** are specified by adding ":" right after the symbol of the element (see *Counterpoise Corrections*).
- **Point charges** are specified with the symbol "Q", followed by the charge (see *Inclusion of Point Charges*).
- **Embedding potentials** are specified by adding a ">" right after the symbol of the element (see *ECP Embedding*).
- **Non-standard** isotopes or nuclear charges are specified with the statements "`M = …`" and "`Z = …`", respectively, after the atomic coordinate definition.

> **Note**
>
> 1. The nuclear charge can adopt non-integer values
> 2. When the nuclear charge is modified throughca "`Z = …`" statement, the total charge of the system should still be calculated based on the unmodified charge. For example, for a calculation of a single hydrogen atom whose `Z` is set to 1.5, a charge of 0 and a spin multiplicity of 2 should be entered into the charge and multiplicity sections of the input file, despite that the actual total charge is 0.5.

- **Fragments** can be conveniently defined by declaring the fragment number a given atom belongs to in parentheses "`(n)`" following the element symbol (see *Fragment Specification*).

2.2.4 Defining Geometry Parameters and Scanning Potential Energy Surfaces

ORCA lets you define the coordinates of all atoms as functions of user defined geometry parameters. By giving not only a value but a range of values (or a list of values) to this parameters potential energy surfaces can be scanned. In this case the variable `RunTyp` is automatically changed to `Scan`. The format for the parameter specification is straightforward:

```
%coords
 CTyp   internal
 Charge 0
 Mult   1
 pardef
  rCH  = 1.09;     # a C-H distance
  ACOH = 120.0;    # a C-O-H angle
  rCO  = 1.35, 1.10, 26; # a C-O distance that will be scanned
 end
 coords
```

(continues on next page)

(continued from previous page)

```
    C  0  0  0   0     0         0
    O  1  0  0  {rCO}  0         0
    H  1  2  0  {rCH}  {ACOH}    0
    H  1  2  3  {rCH}  {ACOH}  180
 end
end
```

In the example above the geometry of formaldehyde is defined in internal coordinates (the geometry functions work exactly the same way with Cartesian coordinates). Each geometric parameter can be assigned as a function of by enclosing an expression within function braces, "`{}` ". For example, a function may look like `*cos(Theta)*rML+R`. Note that all trigonometric functions expect their arguments to be in degrees and not radians. The geometry parameters are expected to be defined such that the lengths come out in Ångströms and the angles in degrees. *After* evaluating the functions, the coordinates will be converted to atomic units. In the example above, the variable `rCO` was defined as a "Scan parameter". Its value will be changed in 26 steps from 1.3 Å down to 1.1 Å and at each point a single point calculation will be done. At the end of the run the program will summarize the total energy at each point. This information can then be copied into the spreadsheet of a graphics program and the potential energy surface can be plotted. Up to three parameters can be scan parameters. In this way grids or cubes of energy (or property) values as a function of geometry can be constructed.

If you want to define a parameter at a series of values rather than evenly spaced intervals, the following syntax is to be used:

```
%coords
 CTyp   internal
 Charge 0
 Mult   1
 pardef
  rCH = 1.09;   # a C-H distance
  ACOH= 120.0;  # a C-O-H angle
  rCO [1.3 1.25 1.22 1.20 1.18 1.15 1.10];  # a C-O distance that will be scanned
 end
 coords
    C  0  0  0   0     0         0
    O  1  0  0  {rCO}  0         0
    H  1  2  0  {rCH}  {ACOH}    0
    H  1  2  3  {rCH}  {ACOH}  180
 end
end
```

In this example the C-O distance is changed in seven non-equidistant steps. This can be used in order to provide more points close to a minimum or maximum and fewer points at less interesting parts of the surface.

A special feature has also been implemented into ORCA - the parameters themselves can be made functions of the other parameters as in the following (nonsense) example:

```
%coords
 CTyp   internal
 Charge 0
 Mult   1
 pardef
  rCOHalf= 0.6;
  rCO    = { 2.0*rCOHalf };
 end
 coords
    C  0  0  0   0     0         0
    O  1  0  0  {rCO}  0         0
    O  1  0  0  {rCO}  180       0
 end
end
```

In this example the parameter `rCO` is computed from the parameter `rCOHalf`. In general the geometry is computed

(assuming a `Scan` calculation) by: (a) incrementing the value of the parameter to be scanned (b) evaluating the functions that assign values to parameters, and (c) evaluating functions that assign values to geometrical variables.

Although it is not mandatory, it is good practice to *first* define the static or scan-parameters and then define the parameters that are functions of these parameters.

Finally, ORCA has some special features that may help to reduce the computational effort for surface scans:

```
%method
 SwitchToSOSCF true       # switches the converger to SOSCF
                          # after the first point. SOSCF may
                          # converge better than DIIS if the
                          # starting orbitals are good.
                          # default = false
 ReducePrint true         # reduce printout after the first point
                          # default=true

 # The initial guess can be changed after the first point.
 # The default is MORead. The MOs of the previous point will,
 # in many cases, be a very good guess for the next point.
 # However, in some cases you may want to be more conservative
 # and use a general guess.

 ScanGuess   OneElec    # the one-electron matrix
             Hueckel    # the extended Hueckel guess
             PAtom      # the PAtom guess
             PModel     # the PModel guess
             MORead     # MOs of the previous point
 end
```

Note

- You can scan along normal modes of a Hessian using the `NMScan` feature as described in section *Normal Mode Scan Calculations Between Different Structures*.
- The surface scan options are also supported in conjunction with TD-DFT/CIS or MR-CI calculations (see section *Potential Energy Surface Scans*).

2.2.5 Mixing internal and Cartesian coordinates

In some cases it may be practical to define some atomic positions in Cartesian and some in internal coordinates. This can be achieved by specifying all coordinates in the `*int` block: using "0 0 0" as reference atoms indicates Cartesian coordinates. Note that for the first atom the flags are "1 1 1", as "0 0 0" would be the normal values for internal coordinates. Consider, for example, the relaxed surface scan from section *Theory*, where the methyl group is given first in an arbitrary Cartesian reference frame and then the water molecule is specified in internal coordinates:

```
! UKS B3LYP SV(P) TightSCF Opt SlowConv
%geom scan B 4 0 = 2.0, 1.0, 15 end end
* int 0 2
# First atom - reference atoms 1,1,1 mean Cartesian coordinates
  C      1   1   1    -0.865590    1.240463   -2.026957

# Next atoms - reference atoms 0,0,0 mean Cartesian coordinates
  H      0   0   0    -1.141534    2.296757   -1.931942
  H      0   0   0    -1.135059    0.703085   -2.943344
  H      0   0   0    -0.607842    0.670110   -1.127819

# Actual internal coordinates
  H      1   2   3     1.999962   100.445    96.050
  O      5   1   2     0.984205   164.404    27.073
```

(continues on next page)

(continued from previous page)

```
  H      6   5   1     0.972562   103.807    10.843
*
```

Internal and Cartesian coordinates can thus be mixed in any order but it is recommended that the first 3 atoms are specified in Cartesian coordinates in order to define a unique reference frame.

2.2.6 Inclusion of Point Charges

In some situations it is desirable to add point charges to the system. In ORCA there are two mechanisms to add point-charges. If you only want to add a few point charges you can "mask" them as atoms as in the following (nonsense) input:

```
# A water dimer
! BP86 def2-SVP

* xyz 0 1
O           1.4190    0.0000    0.0597
H           1.6119    0.0000   -0.8763
H           0.4450    0.0000    0.0898
Q  -0.834  -1.3130    0.0000   -0.0310
Q   0.417  -1.8700    0.7570    0.1651
Q   0.417  -1.8700   -0.7570    0.1651
*
```

Here the "Q"'s define the atoms as point charges. The next four numbers are the magnitude of the point charge and its position. The program will then treat the point charges as atoms with no basis functions and nuclear charges equal to the "Q" values.

If you have thousands of point charges to treat, as in a QM/MM calculation, it is more convenient, and actually necessary, to read the point charges from an external file as in the following example:

```
# A water dimer
! BP86 def2-SVP

% pointcharges "pointcharges.pc"

* xyz 0 1
O    1.4190    0.0000    0.0597
H    1.6119    0.0000   -0.8763
H    0.4450    0.0000    0.0898
*
```

The program will now read the file "`pointcharges.pc`" that contains the point-charge information and then call the module `orca_pc` which adds the point charge contribution to the one-electron matrix and the nuclear repulsion. The file "`pointcharges.pc`" is a simple ASCII file in the following format:

```
3
 -0.834  -1.3130    0.0000   -0.0310
  0.417  -1.8700    0.7570    0.1651
  0.417  -1.8700   -0.7570    0.1651
```

The first line gives the number of point charges. Each consecutive line gives the magnitude of the point charge (in atomic units) and its position (in Ångström units!). However, it should be noted that ORCA treats point charges from an external file differently than "Q" atoms. When using an external point charge file, the interaction between the point charges is not included in the nuclear energy. This behavior originates from QM/MM, where the interactions among the point charges is done by the MM program. These programs typically use an external point charge file when generating the ORCA input. To add the interaction of the point charges to the nuclear energy, the `DoEQ` keyword is used either in the simple input or the `%method` block as shown below.

```
# A non QM/MM pointcharge calculation
! DoEQ

%pointcharges "pointcharges.pc"

%method
   DoEQ true
end
```

Warning

The `DoEQ` keyword has no effect for semi-empirical calculations (e.g., AM1)!

2.3 Basic Calculation Settings

2.3.1 Run Types

The type of calculation to be performed can be chosen via the simple input – for example, `! Opt` – or via the `RunTyp` variable in `%method`:

```
%method
  RunTyp  Gradient  # Single point energy and gradient
end
```

The full list of available run types is given in Section 2.3.3. Note that some run types are triggered automatically under certain conditions and cannot be requested explicitly via the `RunTyp` keyword.

2.3.2 Method Classes

ORCA provides several classes of methods: Hartree–Fock (HF), Density functional theory (DFT), semi-empirical methods based on the NDDO approach, etc. This choice is controlled via the simple input, e.g. `!HF`, or via the `method` variable in the `%method` block:

```
%method
  method DFT
end
```

The list of available methods is given in Section 2.3.3.

2.3.3 Keywords

Table 2.4: Simple input keywords for basic calculation settings.

Keyword	Description
Run Types	
Energy	Single point energy calculation (Default. Aliases: SinglePoint, SP)
EnGrad	Single point energy and gradient calculation (Alias: EnergyGrad)
Opt	*Geometry optimization*, see also Table 4.1
MD	*Molecular dynamics simulation*
CIM	*Cluster-in-molecules* calculation
PrintThermoChem	*Thermochemistry calculation* from previously calculated Hessian
PropertiesOnly	*Compute properties* from previously calculated densities
Docker	*Host-guest docking*
EDA	*Energy decomposition analysis*
NMScan	**Deprecated!** Normal mode scan (Alias: NormalModeScan)
NMGrad	**Deprecated!** Normal mode gradient (Alias: NMGradient)
MTR	**Deprecated!** Mode trajectory (Aliases: MT, ModeTrajectory)
Methods	
HF	*Hartree–Fock*; RHF, UHF, and ROHF also set the *wavefunction type*
DFT	*Density functional theory*; RKS, UKS, and ROKS also set the *wavefunction type*
<NDO-Method>	*NDO*-based method keywords are given in Table 3.22
<XTB-Method>	*Native xTB*-based method keywords are given in Table 3.27
MM	*Molecular mechanics*

Table 2.5: %method block input keywords for basic calculation settings.

Keyword	Options	Description
RunTyp	Energy	Single point energy calculation (Default. Aliases: SinglePoint, SP)
	Gradient	Single point energy and gradient calculation
	Opt	*Geometry optimization* (Aliases: Geom, Geometry, GeometryOpt)
	Scan	*Unrelaxed geometry parameter scan* (Alias: Trajectory)
	CIM	*Cluster-in-molecules* calculation
	NMScan	**Deprecated!** Normal mode scan (Alias: NormalModeScan)
	NMGrad	**Deprecated!** Normal mode gradient (Aliases: NMGradient, NormalModeGradient)
	MTR	**Deprecated!** Mode trajectory (Aliases: MT, ModeTrajectory)
Method	HF	(Default) *Hartree–Fock*
	DFT	*Density functional theory*
	<NDO-Method>	*NDO*-based method options are given in Table 3.23

2.4 Control of Output

ORCA provides various options to control the amount of printed output. This is specifically important as some methods and protocols can create very large amount of output and data. General control of the output is provided by the PrintLevel and Print[<option>] keywords in the %output block: PrintLevel can be used to select certain default settings for the print array. Specifying Print after PrintLevel can be used to modify these defaults.

```
%output
  PrintLevel      Normal
  Print[ Flag ]   0  # turn print off
                  1  # turn print on
                  n  # some flags are more sophisticated
end
```

2.4.1 Print Options

A detailed list of options for `PrintLevel` and `Print` is given in table Table 2.6. A list of option toggled by each printlevel can be found in Table 2.7.

Table 2.6: `%output` block input keywords and options for print control.

Keyword	Option	Description
`PrintLevel`	`nothing`	Deactivates output printing
	`mini`	Sets printlevel to mini
	`small`	Sets printlevel to small
	`normal`	Sets printlevel to normal (default)
	`maxi`	Sets printlevel to maxi
	`large`	Sets printlevel to large
	`huge`	Sets printlevel to huge
	`debug`	Sets printlevel to debug
`Print[ <option> ]`	`P_InputFile`	Echo the input file
	`P_Cartesian`	Print the cartesian coordinates
	`P_Internal`	Print the internal coordinates
	`P_Basis`	$= 1$: Print the basis set information
		$= 2$: Also print the primitives in input format
	`P_OneElec`	Print of the one electron matrix
	`P_Overlap`	Print the overlap matrix
	`P_KinEn`	Print the kinetic energy matrix
	`P_S12`	Print the $S^{-1/2}$ matrix
	`P_GuessOrb`	Print the initial guess orbitals
	`P_OrbEn`	$= 1$: Print orbital energies up to LUMO+9
		$= 2$: Print all orbital energies
	`P_MOs`	Print the MO coefficients on convergence
	`P_Density`	Print the converged electron density
	`P_SpinDensity`	Print the converged spin density
	`P_EHTDetails`	Print initial guess extended Hückel details
	`P_SCFInfo`	Print the SCF input flags
	`P_SCFMemInfo`	Print the estimated SCF memory requirements
	`P_SCFIterInfo`	$= 1$: Print short iteration information
		$= 2$: Print longer iteration information
		$= 3$: In a direct SCF also print integral progress
	`P_Fockian`	Print Fockian matrix
	`P_DIISMat`	Print DIIS matrix
	`P_DIISError`	Print DIIS error
	`P_Iter_P`	Print Density
	`P_Iter_C`	Print MO coefficients
	`P_Iter_F`	Print Fock matrix
	`P_Mayer`	Print Mayer population analysis (Default = on)
	`P_NatPop`	Print Natural population analysis (Default = off)
	`P_NPA`	Print Natural population analysis (Default = off)
	`P_Hirshfeld`	Print Hirshfeld population analysis (Default = off)
	`P_MBIS`	Print MBIS population analysis (Default = off)
	`P_Mulliken`	Print Mulliken population analysis (Default = on)
	`P_AtCharges_M`	Print Mulliken atomic charges
	`P_OrbCharges_M`	Print Mulliken orbital charges
	`P_FragCharges_M`	Print Mulliken fragment charges
	`P_FragBondOrder_M`	Print Mulliken fragment bond orders
	`P_BondOrder_M`	Print Mulliken bond orders
	`P_ReducedOrbPop_M`	Print Mulliken reduced orbital charges
	`P_FragPopMO_M`	Print Mulliken fragment population for each MO
	`P_FragOvlMO_M`	Print Mulliken overlap populations per fragment pair

continues on n

Table 2.6 – continued from previous page

Keyword	Option	Description
	P_AtPopMO_M	Print Mulliken atomic charges in each MO
	P_OrbPopMO_M	Print Mulliken orbital population for each MO
	P_ReducedOrbPopMO_M	Print Mulliken reduced orbital population for each MO
	P_Loewdin	Print Loewdin population analysis (Default = on)
	P_AtCharges_L	Print Loewdin atomic charges
	P_OrbCharges_L	Print Loewdin orbital charges
	P_FragCharges_L	Print Loewdin fragment charges
	P_FragBondOrder_L	Print Loewdin fragment bond orders
	P_BondOrder_L	Print Loewdin bond orders
	P_ReducedOrbPop_L	Print Loewdin reduced orbital charges
	P_FragPopMO_L	Print Loewdin fragment population for each MO
	P_FragOvlMO_L	Print Loewdin overlap populations per fragment pair
	P_AtPopMO_L	Print Loewdin atomic charges in each MO
	P_OrbPopMO_L	Print Loewdin orbital population for each MO
	P_ReducedOrbPopMO_L	Print Loewdin reduced orbital population for each MO
	P_NPA	Natural population analysis
	P_Fragments	Print fragment information
	P_GUESSPOP	Print initial guess populations
	P_UNO_FragPopMO_M	Print Mulliken fragment population per UNO
	P_UNO_OrbPopMO_M	Print Mulliken orbital pop. per UNO
	P_UNO_AtPopMO_M	Print Mulliken atomic charges per UNO
	P_UNO_ReducedOrbPopMO_M	Print Mulliken reduced orbital pop. per UNO
	P_UNO_FragPopMO_L	Print Loewdin fragment population per UNO
	P_UNO_OrbPopMO_L	Print Loewdin orbital pop. per UNO
	P_UNO_AtPopMO_L	Print Loewdin atomic charges per UNO
	P_UNO_ReducedOrbPopMO_L	Print Loewdin reduced orbital pop. per UNO
	P_UNO_OccNum	Print occupation numbers per UNO
	P_AtomExpVal	Print atomic expectation values
	P_AtomBasis	Print atomic basis
	P_AtomDensFit	Print electron density fit
	P_Symmetry	Symmetry basic information
	P_Sym_Salc	Symmetry process printing
	P_SCFSTABANA	Information on progress, convergence, and results of SCF stabilit
	P_DFTD	Print info on Grimme's dispersion correction
	P_DFTD_GRAD	Print gradient info on Grimme's dispersion correction
	P_G1EL2EL	Print one- and two-electron contributions to g-tensor

The various choices for PrintLevel have the following defaults:

Table 2.7: Default print options invoked by the respective `PrintLevel`.

PrintLevel	Print options included	
mini	P_OrbEn	= 1
	P_Cartesian	= 1
	P_InputFile	= 1
	P_SCFIterInfo	= 1
small	all the previous plus	
	P_SCFInfo	= 1
	P_Mayer	= 1
	P_MULLIKEN	= 1
	P_AtCharges_M	= 1
	P_ReducedOrbPop_M	= 1
	P_Loewdin	= 1
	P_AtCharges_L	= 1

continues on next page

Table 2.7 – continued from previous page

PrintLevel	Print options included	
	P_ReducedOrbPop_L	= 1
	P_Fragments	= 1
	P_FragCharges_M	= 1
	P_FragBondOrder_M	= 1
	P_FragCharges_L	= 1
	P_FragBondOrder_L	= 1
	P_DFTD	= 1
normal	all the previous plus	
	P_Internal	= 1
	P_BondOrder_L	= 1
	P_BondOrder_M	= 1
	P_FragPopMO_L	= 1
	P_ReducedOrbPopMO_L	= 1
	P_SCFIterInfo	= 2
maxi	all the previous plus	
	P_OrbEn	= 2
	P_GuessOrb	= 1
	P_MOs	= 1
	P_Density	= 1
	P_SpinDensity	= 1
	P_Basis	= 1
	P_FragOVLMO_M	= 1
	P_OrbPopMO_M	= 1
	P_OrbCharges_M	= 1
	P_DFTD	= 2
	P_DFTD_GRAD	= 1
huge	All the previous plus	
	P_OneElec	= 1
	P_Overlap	= 1
	P_S12	= 1
	P_AtPopMO_M	= 1
	P_OrbPopMO_M	= 1
	P_AtPopMO_L	= 1
	P_EHTDetails	= 1
	P_DFTD_GRAD	= 2
debug	Prints everything	

2.4.2 Simple Input Keywords

For convenience, a variety of simple input keywords that control specific print settings and additional outputs have been implemented as well (cf. Table 2.8).

Table 2.8: Simple input keywords for specific output control.

Keyword	Print options toggled	Description
`MiniPrint`	`PrintLevel mini`	Activates printlevel mini
`SmallPrint`	`PrintLevel small`	Activates printlevel small (default)
`NormalPrint`	`PrintLevel normal`	Activates printlevel normal
`LargePrint`	`PrintLevel large`	Activates printlevel mini
`PrintMOs`	`Print[ p_MOs ]`	Prints MO coefficients
`NoPrintMOs`		Suppress printing of MO coefficients
`PrintBasis`	`Print[ p_basis ]`	Print the basis set in input format
`PrintGap`	`Print[ p_homolumogap ]`	Prints the HOMO/LUMO gap in each SCF iteration.
`ReducedPop`		Prints Loewdin reduced orb.pop per MO
`NoReducedPop`		Deactivates printing of Loewdin reduced orb.pop per MO
`AIM`		Produce a WFN file
`XYZFILE`		Produce an XYZ coordinate file
`PDBFILE`		Produce a PDB file
`UNO`		Produce a *UHF natural orbitals*
`NoPropFile`		Do not write to the *property file*
`KeepTransDensity`		Keep the transition density matrices on disk

2.5 Parallel and Multi-Process Runs

Most of the important modules in ORCA can run in parallel or in multi-process mode: There are parallel versions for Linux, MAC and Windows computers which make use of OpenMPI (open-source MPI implementation) and Microsoft MPI (Windows only). Parallel execution means that the different processes perform the task in synchronous manner, communicating results and synchronizing execution (via MPI). The multi-process mode also employs multiple processes. But these work independently, not knowing - and not needing to know what the other processes are doing. The latest ORCA version even can combine both modes. Please see the remarks in *Multi-Process Calculations* for more details.

Parallel (or multi-process) execution is requested in the input via

```
! PAL4  # everything from PAL2 to PAL8 and Pal16, Pal32, Pal64 is recognized
```

or

```
%pal nprocs 4 end # any number (positive integer)
```

Assuming that the MPI libraries are properly installed on your computer, it is fairly easy to run the parallel version of ORCA: You simply specify the number of parallel processes in the input and call (serial) ORCA (with full path!) The parallelized modules of ORCA are started by the (serial) ORCA-Driver. If the driver finds `PAL4` or `%pal nprocs 4 end` (e.g.) in the input, it will start up the parallel modules instead of the serial ones.

Warning

Do not start the ORCA driver with mpirun!

(Please see section *Hints on the Use of Parallel ORCA* for what else has to be taken care of for a successfull parallel run of ORCA.)

2.5.1 Parallel and Multi-Process Modules

The following modules and utility programs are presently parallelized - or usable in Multi-Process Mode:

List of Parallelized Modules

AUTOCI - all methods
CASSCF / NEVPT2 / CASPT2 / CASSCFRESP
CIPSI
CIS/TDDFT / CISRESP
EDA
GRAD
GUESS
LEANSCF
MAGRELAX
MCRPA
MDCI (Canonical- and DLPNO-Methods)
MM
MP2 and RI-MP2 (including Gradients)
MRCI
PC
PLOT
PNMR
POP
PROP
PROPINT
REL
ROCIS
SCFGRAD
SCFRESP (with SCFHessian)
STARTUP
VPOT

(For a complete list of all modules and the description of their functionality, please refer to *Program Components*)

The efficiency of the parallel modules is such that for RI-DFT perhaps up to 16 processors are a good idea while for hybrid DFT and Hartree-Fock a few more processors are appropriate. Above this, the overhead becomes significant and the parallelization loses efficiency. Coupled-cluster calculations usually scale well up to at least 8 processors but probably it is also worthwhile to try 16.

List of Multi-Process Modules

Numerical Gradients, Frequencies, Overtones-and-Combination-Bands
VPT2
NEB (Nudged Elastic Band)
GOAT (Global Optimizer Algorithm)

For Numerical Frequencies or Gradient runs it makes sense to choose nprocs = 4 or 8 times 6*Number of Atoms. For VPT2 on larger systems you may well even try 16 times 6*Number of Atoms - if you use multiple processes per

displacement. (Please check out the section *Hints on the Use of Parallel ORCA* what you have to take care of for such kind of calculations.)

> **Note**
>
> Parallelization is a difficult undertaking and there are many different protocols that work differently for different machines. Please understand that we can not provide support for each and every platform. We are trying our best to make the parallelization transparent and provide executables for various platforms but we can not guarantee that they always work on every system. Please see the download information for details of the version.

2.5.2 Hints on the Use of Parallel ORCA

Many questions that are asked in the discussion forum deal with the parallel version of ORCA. Please understand that we cannot possibly provide one-on-one support for every parallel computer in the world. So, please make every effort to solve the technical problems locally together with your system administrator. Here are some explanations about what is special to the parallel version, which problems might arise from this and how to deal with them:

Single Node Runs

1. Parallel ORCA can be used with OpenMPI (on Linux and MAC) or MS-MPI (on windows) only. Please see the download information for details of the relevant OpenMPI-version for your platform.

   The OpenMPI version is configurable in a large variety of ways, which cannot be covered here. For a more detailed explanation of all available options, cf. http://www.open-mpi.org

2. Please note that the OpenMPI version is dynamically linked, that is, it needs at runtime the OpenMPI libraries (and several other standard libraries)! If you compile MPI on your own computer, you also need to have a fortran compiler, as `mpirun` will contain fortran bindings.

   (Remember to set `PATH` and `LD_LIBRARY_PATH` to mpirun and the mpi libraries)

3. Many problems arise, because ORCA does not find its parallel executables. To avoid this, it is crucial to call ORCA with its complete pathname. The easiest and safest way to do so is to include the directory with the ORCA-executables in your `$PATH`. Then start the calculation:

   ```
   - interactively:
     start orca with full path: "/mypath_orca_executables/orca MyMol.inp"
   - batch :
     set your path: `export PATH=/mypath_orca_executables:$PATH` (for bash) then
     start orca with full path: "$PATH/orca $jobname.inp"
   ```

   This seems redundant, but it really is important if you want to run a parallel calculation to call ORCA with the full path! Otherwise it will not be able to find the parallel executables.

4. It is recommended to run orca in local (not nfs-mounted) scratch-directories, (`/tmp1` or `/usr/local` e.g.) and to renew these directories for each run to avoid confusion with left-overs of a previous run.

5. It has proven convenient to use "wrapper" scripts. These scripts should

   ```
   - set the path
   - create local scratch directories
   - copy input files to the scratch directory
   - start orca
   - save your results
   - remove the scratch directory
   ```

   A basic example of such a submission script for the parallel ORCA version is shown later (13.) (this is for the Torque/PBS queuing system, running on Apple Mac OS X):

6. Parallel ORCA distinguishes the following cases of disk availability:

1. each process works on its own (private) scratch directory (the data on this directory cannot be seen by any other process). This is flagged by **"working on local scratch directories"**
2. all processes work in a common scratch directory (all processes can see all file-data) ORCA will distinguish two situations:
   - all processes are on the same node - flagged by **"working on a common directory"**
   - the processes are distributed over multiple nodes but accessing a shared filesysten - flagged by **"working on a shared directory"**
3. there are at least 2 groups of processes on different scratch directories, one of the groups consisting of more than 1 process - flagged by **"working on distributed directories"**

Parallel ORCA will find out, which of these cases exists and will handle the I/O respectively. If ORCA states disk availability differently from what you would expect, check the number of available nodes and/or the distribution pattern (fill_up/round_robin)

7. It is possible to pass additional MPI-parameters to mpirun by adding these arguments to the ORCA call - all arguments enclosed in a single pair of quotes:

```
/mypath_orca_executables/orca MyMol.inp "--bind-to core"
```

– or – for multiple arguments

```
/mypath_orca_executables/orca MyMol.inp "--bind-to core --verbose"
```

Multi-Node Runs - Remote Execution

1. If Parallel ORCA finds a file named "`MyMol.nodes`" in the directory where it's running, it will use the nodes listed in this file to start the processes on, provided your input file was "`MyMol.inp`". You can use this file as your machinefile specifying your nodes, using the usual OpenMPI machinefile notation.

```
node1 cpu=2
node2 cpu=2
```

or

```
node1
node1
node2
node2
```

If you run the Parallel ORCA version on only one computer, you do not need to provide a nodefile, and neither have to enable an rsh/ssh access, as in this case the processes will simply be forked! If you start ORCA within a queueing system, you also don't need to provide a nodefile. The queueing system will care for it.

2. If the ORCA-environment variables are not equally defined on all participating compute nodes it might be advisable to export these variables. This can be achieved by passing the following additional parameters to mpirun via the ORCA call:

```
/mypath_orca_executables/orca MyMol.inp "-x LD_LIBRARY_PATH -x PATH"
```

3. OpenMPI requires that the PATH environment variable be set to find executables on remote nodes. As it is not always possible to change the startup scripts accordingly, OpenMPI provides the additional option: "`--prefix`" Calling ORCA and requesting multiple processes for the parallelized modules should then look like

```
/mypath_orca_executables/orca MyMol.inp "--prefix /my-openmpi-folder"
/mypath_orca_executables/orca MyMol.inp "--prefix /my-openmpi-folder --
↪machinefile MyMol.nodes" (if not started via queueing system)
```

As ORCA is dynamically linked it also needs to know the location of the ORCA libraries This can be communicated via

```
/mypath_orca_executables/orca MyMol.inp "--prefix /my-openmpi-folder --
↪machinefile MyMol.nodes -x LD_LIBRARY_PATH"
```

Multi-Process Calculations

1. An additional remark on multi-process numerical calculations (Frequencies, Gradient, (Hybrid) Hessian), VPT2, NEB, GOAT:
   The processes that execute these calculations do not work in parallel, but independently, often in a totally asynchronous manner. The numerical calculations will start as many processes, as you dedicated for the parallel parts before and they will run on the same nodes. If your calculation runs on multiple nodes, you have to set the environment variable `RSH_COMMAND` to either "`rsh`" or "`ssh`". If `RSH_COMMAND` is not defined, ORCA will abort. This prevents that all processes of a multi-node run are started on the 'master'-node.

2. On multiple user request the 'parallelization' of NumCalc has been made more flexible. If before ORCA would start nprocs displacements with a single process each, the user can now decide on how many processes should work together on a single displacement.

   For this the nprocs keyword got a sibling:

```
%pal nprocs       32 # or nprocs_world - total number of parallel processes
     nprocs_group  4 #                 - number of parallel processes per sub-
↪task
     end
```

   This setting will ORCA make use 32 processes, with 4 processes working on the same displacement, thus running 8 displacements simultaneously. The methods that can profit from this new feature are

   - all NumCalc-methods: as NumGrad, NumFreq, VPT2, Overtones, NEB, and GOAT.
   - the analytical (parallel) Hessian, leading to a nice increase of parallel performance for really large calculations.

   It is highly recommended to choose nprocs_group to be an integer divisor of nprocs_world!

   For convenient use a couple of standard 'groupings' are made available via simple input keyword:

```
!PAL4(2x2)   # 2 groups a 2 workers
!PAL8(4x2)   # 4 groups a 2 workers
!PAL8(2x4)   # ...
!PAL16(4x4)
!PAL32(8x4)
!PAL32(4x8)
!PAL64(8x8)
```

   Note

   If your system-administration does not allow to connect via rsh/ssh to other compute nodes, you unfortunately cannot make use of parallel sub-calculations within NumCalc runs. This affects NEB as well as GOAT, VPT2, Overtone-and-Combination-Bands, as well as Numerical Frequencies and Gradients.

3. Wrapper script to start ORCA

```
#!/bin/zsh

setopt EXTENDED_GLOB
setopt NULL_GLOB
#export MKL_NUM_THREADS=1
```

(continues on next page)

(continued from previous page)

```
b=${1:r}

#get number of procs.... close your eyes... (it really works!)
if [[ ${$(grep -e '^!' $1):u} == !*(#b)PAL(<0-9>##)* ]]; then
  nprocs=$match
let "nodes=nprocs"
elif [[ ${(j: :) $(grep -v '^#' $1):u} == *%(#b)PAL*NPROCS' '#(<0-9>##)* ]];␣
↪then
  nprocs=$match
  let "nodes=nprocs"
fi

cat > ${b}.job <<EOF
#!/bin/zsh
#PBS -l nodes=1:ppn=${nodes:=1}
#PBS -S /bin/zsh
#PBS -l walltime=8760:00:00

setopt EXTENDED_GLOB
setopt NULL_GLOB
export PATH=$PBS_O_PATH

logfile=$PBS_O_WORKDIR/${b}.log
tdir=$(mktemp -d /Volumes/scratch/$USER/${b}__XXXXXX)

trap '
echo "Job terminated from outer space!" >> $logfile
rm -rf $tdir
exit
' TERM

cp $PBS_O_WORKDIR/$1 $tdir
foreach f ($PBS_O_WORKDIR/*.gbw $PBS_O_WORKDIR/*.pot) { cp $f $tdir }
cd $tdir

echo "Job started from ${PBS_O_HOST}, running on $(hostname) in $tdir using
$(which orca)" > $log
file
=orca $1 1>>$logfile 2>&1

cp ^(*.(inp|tmp*))  $PBS_O_WORKDIR/
rm -rf $tdir

EOF

qsub -j oe -o ${b}.job.out ${b}.job
```

2.6 Self-Consistent-Field (SCF)

SCF convergence is a pressing problem in any electronic structure package because the total execution times increases linearly with the number of iterations. Thus, it remains true that the best way to enhance the performance of an SCF program is to make it converge better. In some cases, especially for open-shell transition metal complexes, convergence may be very difficult. ORCA makes a dedicated effort to achieve reasonable SCF convergence for these cases without compromising efficiency.

Another issue is whether the solution found by ORCA is stable, i.e. a minimum on the surface of orbital rotations. Especially for open-shell singlets it can be hard to achieve a broken-symmetry solution. The SCF stability analysis (section *SCF Stability Analysis*) may be able to help in such situations. Please also note that if `! TRAH` is used the

solution must be a true local minimum though not necessarily a global.

Tip

The expectation value $\langle S^2 \rangle$ is an estimation of the spin contamination in the system. It is highly recommended for open-shell systems, especially with transition metal complexes, to check the UCO (unrestricted corresponding orbitals, see *UNO, UCO and QROs input*) overlaps and visualise the corresponding orbitals. Additionally, spin-population on atoms that contribute to the singly occupied orbitals is also an identifier of the electronic structure.

2.6.1 Convergence Tolerances

Before discussing how to converge a SCF calculation it should be defined what is meant by "converged". ORCA has a variety of options to control the target precision of the energy and the wavefunction that can be selected in the `% scf` block, or with a simple input line keyword that merges the criterion label with "SCF", e.g. `! StrongSCF` or `! VeryTightSCF`:

```
%scf
  Convergence             # The default convergence is between medium and strong
               Sloppy     # very weak convergence
               Loose      # still weak convergence
               Medium     # intermediate accuracy
               Strong     # stronger
               Tight      # still stronger
               VeryTight  # even stronger
               Extreme    # close to numerical zero of the computer
                          # in double precision arithmetic
 end
```

Like other keys, `Convergence` is a compound key that assigns default values to a variety of other variables given in the box below. In Table 2.9 we present the chosen values for each compound key. If the corresponding simple inputs are given (`!StrongSCF`, `!VeryTightSCF`, …etc), then in addition the values in Table 2.10 are also set, which also influence the behavior of post-SCF modules. The default convergence criteria are reasonable and should be sufficient for most purposes. For a cursory look at populations weaker convergence may be sufficient, whereas other cases may require stronger than default convergence. Note that `Convergence` does not only affect the target convergence tolerances but also the integral accuracy as discussed in the section about direct SCF and alike. **This is very important: if the error in the integrals is larger than the convergence criterion, a direct SCF calculation cannot possibly converge.**

The convergence criteria are always printed in the output. Given below is a list of the convergence criteria for `! TightSCF`, which is often used for calculations on transition metal complexes.

```
%scf
  TolE        1e-8  # energy change between two cycles
  TolRMSP     5e-9  # RMS density change
  TolMaxP     1e-7  # maximum density change
  TolErr      5e-7  # DIIS error convergence
  TolG        1e-5  # orbital gradient convergence
  TolX        1e-5  # orbital rotation angle convergence
  Thresh    2.5e-11 # integral prescreening threshold
  TCut      2.5e-12 # primitive integral prescreening cutoff
  ConvCheckMode  2  # = 0: check all convergence criteria
                    # = 1: stop if one of criterion is met, this is sloppy!
                    # = 2: check change in total energy and in one-electron energy
                    #      Converged if delta(Etot)<TolE and delta(E1)<1e3*TolE
  ConvForced        # = 0: convergence not mandatory for next calculation step
                    # = 1: break, if you did not meet the convergence criteria
end
%method
```

(continues on next page)

(continued from previous page)

```
  Z_Tol       1e-4  # CP-SCF solver tolerance
  BFCut       1e-11 # basis function cutoff for numerical integration
end

# Additional tolerances modified by the simple input
%casscf
  GTol       2.5e-4
  ETol       2.5e-8
end
%mdci
  STol         1e-5
end
%autoci
  STol         1e-5
end
%mrci
  ETol       2.5e-7
  RTol       2.5e-7
end
%cis
  ETol       2.5e-7
  RTol       2.5e-7
end
```

Table 2.9: Threshold choices for convergence keywords.

Variable	Value
Sloppy (`!SloppySCF/Convergence Sloppy`)	
`TolE`	3e-5
`TolMAXP`	1e-4
`TolRMSP`	1e-5
`TolErr`	1e-4
`Thresh`	1e-9
`TCut`	1e-10
`BFCut`	1e-10
`TolG`	3e-4
`TolX`	3e-4
`Z_Tol`	5e-3
Loose (`!LooseSCF/Convergence Loose`)	
`TolE`	1e-5
`TolMAXP`	1e-3
`TolRMSP`	1e-4
`TolErr`	5e-4
`Thresh`	1e-9
`TCut`	1e-10
`BFCut`	1e-10
`TolG`	1e-4
`TolX`	1e-4
`Z_Tol`	3e-3
Medium (`!MediumSCF/Convergence Medium`)	
`ConvCheckMode`	2
`TolE`	1e-6
`TolMAXP`	1e-5
`TolRMSP`	1e-6
`TolErr`	1e-5
`Thresh`	1e-10
`TCut`	1e-11

continues on next page

Table 2.9 – continued from previous page

Variable	Value
BFCut	1e-10
TolG	5e-5
TolX	5e-5
Z_Tol	1e-3
Strong (!StrongSCF/Convergence Strong)	
ConvCheckMode	2
TolE	3e-7
TolMAXP	3e-6
TolRMSP	1e-7
TolErr	3e-6
Thresh	1e-10
TCut	3e-11
BFCut	3e-11
TolG	2e-5
TolX	2e-5
Z_Tol	7e-4
Tight (!TightSCF/Convergence Tight)	
ConvCheckMode	2
TolE	1e-8
TolMAXP	1e-7
TolRMSP	5e-9
TolErr	5e-7
Thresh	2.5e-11
TCut	2.5e-12
BFCut	1e-11
TolG	1e-5
TolX	1e-5
Z_Tol	1e-4
VeryTight (!VeryTightSCF/Convergence VeryTight)	
ConvCheckMode	2
TolE	1e-9
TolMAXP	1e-8
TolRMSP	1e-9
TolErr	1e-8
Thresh	1e-12
TCut	1e-14
BFCut	1e-12
TolG	2e-6
TolX	2e-6
Z_Tol	3e-5
Extreme (!ExtremeSCF/Convergence Extreme)	
ConvCheckMode	0
TolE	1e-14
TolMAXP	1e-14
TolRMSP	1e-14
TolErr	1e-14
Thresh	3e-16
TCut	3e-16
TolG	1e-09
TolX	1e-09
Z_Tol	3e-06
BFCut	3e-16

Table 2.10: Additional threshold choices set by the simple input keys (`!StrongSCF`, …etc.)

Block	Variable	Value
Sloppy (`!SloppySCF`)		
casscf	GTol	5.0e-3
	ETol	1.0e-6
mdci	STol	1.0e-4
mrci	ETol	1.0e-5
	RTol	1.0e-5
cis	ETol	1.0e-5
	RTol	1.0e-5
Loose (`!LooseSCF`)		
casscf	GTol	5.0e-3
	ETol	1.0e-6
mdci	STol	1.0e-4
autoci	STol	1.0e-4
mrci	ETol	1.0e-5
	RTol	1.0e-5
cis	ETol	1.0e-5
	RTol	1.0e-5
Normal (`!NormalSCF`)		
casscf	GTol	1.0e-3
	ETol	1.0e-7
mdci	STol	2.5e-5
autoci	STol	2.5e-5
mrci	ETol	1.0e-6
	RTol	1.0e-6
cis	ETol	1.0e-6
	RTol	1.0e-6
Strong (`!StrongSCF`)		
casscf	GTol	5.00e-4
	ETol	6.66e-8
mdci	STol	7.50e-6
autoci	STol	7.50e-6
mrci	ETol	6.66e-7
	RTol	6.66e-7
cis	ETol	6.66e-7
	RTol	6.66e-7
Tight (`!TightSCF`)		
casscf	GTol	2.5e-4
	ETol	2.5e-8
mdci	STol	1.0e-5
autoci	STol	1.0e-5
mrci	ETol	2.5e-7
	RTol	2.5e-7
cis	ETol	2.5e-7
	RTol	2.5e-7
VeryTight (`!VeryTightSCF`)		
casscf	GTol	1.0e-5
	ETol	1.0e-8
mdci	STol	1.0e-6
autoci	STol	1.0e-6
	D3Thresh	1.0e-14
	D4Thresh	1.0e-14
	D5Thresh	1.0e-14
mrci	ETol	1.0e-7
	RTol	1.0e-7

continues on next page

Table 2.10 – continued from previous page

Block	Variable	Value
cis	ETol	1.0e-7
	RTol	1.0e-7
Extreme (!ExtremeSCF)		
casscf	GTol	1.0e-9
	ETol	1.0e-12
mdci	STol	1.0e-9
	TCutInt	0.0
autoci	STol	1.0e-9
	D3Thresh	1.0e-14
	D4Thresh	1.0e-14
	D5Thresh	1.0e-14
mrci	ETol	1.0e-12
	RTol	1.0e-12
cis	ETol	1.0e-12
	RTol	1.0e-12

There is an additional set of simple keywords, !SCFCONV<n> with n = 6–12 (e.g. !SCFCONV8), which sets TolE = 10^{-n} and *most* of the other tolerances listed above to appropriate values. However, some, like Z_Tol, TolRMSP, and TolMAXP are not adjusted! Hence, it is recommended to use the keywords given above instead.

If ConvCheckMode=0, *all* convergence criteria have to be satisfied for the program to accept the calculation as converged, which is a quite rigorous criterion. In this mode, the program also has mechanisms to decide that a calculation is converged even if one convergence criterion is not fulfilled but the others are overachieved. ConvCheckMode=1 means that one criterion is enough. This is quite dangerous, so ensure that none of the criteria are too weak, otherwise the result will be unreliable. The default ConvCheckMode=2 is a check of medium rigor — the program checks for the change in total energy and for the change in the one-electron energy. If the ratio of total energy and one-electron energy is constant, the self-consistent field does not fluctuate anymore and the calculation can be considered converged. If you have small eigenvalues of the overlap matrix, the density may not be converged to the number of significant figures requested by TolMaxP and TolRMSP.

ConvForced is a flag to prevent time consuming calculations on non-converged wave functions. It will default to ConvForced=1 for Post-HF methods, Excited States runs and Broken Symmetry calculations. You can overwrite this default behavior by setting ConvForced=0.

Irrespective of the ConvForced value that has been chosen, properties or numerical calculations (NumGrad, NumFreq) will not be performed on non-converged wavefunctions!

2.6.2 Dynamic and Static Damping

Damping is the oldest and simplest convergence aid. It was already invented by Douglas Hartree when he did his famous atomic calculations. Damping consists of mixing the old density with the new density as:

$$P_{\text{new, damped}} = (1-\alpha)\, P_{\text{new}} + \alpha P_{\text{old}} \tag{2.1}$$

where α is the damping factor, which must have a value of less than 1. Thus the permissible range (not checked by the program) is 0 …0.999999. For α values larger than 1, the calculation cannot proceed since no new density is admixed. Damping is important in the early stages of a calculation where P_{old} and P_{new} are very different from each other and the energy is strongly fluctuating. Many schemes have been suggested that vary the damping factor dynamically to give strong damping in the beginning and no damping in the end of an SCF. The scheme implemented in ORCA is that by Hehenberger and Zerner [1] and is invoked with CNVZerner=true. Static damping is invoked with CNVDamp=true. These convergers are mutually exclusive. They can be used in the beginning of a calculation when it is not within the convergence radius of DIIS or SOSCF. Damping works reasonably well, but most other convergers in ORCA are more powerful.

If damping used in conjunction with DIIS or SOSCF, the value of DampErr is important: once the DIIS error falls below DampErr, the damping is turned off. In case SOSCF is used, DampErr refers to the orbital gradient value

at which the damping is turned off. The default value is 0.1 Eh. In difficult cases, however, it is a good idea to choose `DampErr` much smaller, e.g. 0.001. This is — to some extent — chosen automatically together with the keyword `! SlowConv`.

```
%scf
  # control of the Damping procedure
  CNVDamp   true    # default: true
  CNVZerner false   # default: false
  DampFac   0.98    # default: 0.7
  DampErr   0.05    # default: 0.1
  DampMin   0.1     # default: 0.0
  DampMax   0.99    # default: 0.98
  # more convenient:
  Damp fac 0.98 ErrOff 0.05 Min 0.1 Max 0.99 end
end
```

2.6.3 Level Shifting

Level shifting is a frequently used technique. The basic idea is to shift the energies of the virtual orbitals such that after diagonalization the occupied and virtual orbitals mix less strongly and the calculation converges more smoothly towards the desired state. Also, level shifting should prevent flipping of electronic states in near-degenerate cases. In a special context it has been shown by Saunders and Hillier [2, 3] to be equivalent to damping.

Similar to `DampErr` described in the previous section, `ShiftErr` refers to the DIIS error at which the level shifting is turned off.

```
%scf
  # control of the level shift procedure
  CNVShift   true  # default: true
  LShift     0.1   # default: 0.25, energy unit is Eh.
  ShiftErr   0.1   # default: 0.0
  # more convenient:
  Shift Shift 0.1 ErrOff 0.1 end
end
```

2.6.4 Direct Inversion in Iterative Subspace (DIIS)

The direct inversion in iterative subspace (DIIS) is a technique that was invented by Pulay [4, 5]. It has become the *de facto* standard in most modern electronic structure programs, because DIIS is robust, efficient and easy to implement. Basically DIIS uses a criterion to judge how far a given trial density is from self-consistency. The commutator of the Fock and density matrices [**F**,**P**] is a convenient measure for this error. With this information, an extrapolated Fock matrix from the present and previous Fock matrices is constructed, which should be much closer to self-consistency. In practice this is usually true, and better than linear convergence has been observed with DIIS. In some rare (open-shell) cases however, DIIS convergence is slow or absent after some initial progress. As self-consistency is approached, the set of linear equations to be solved for DIIS approaches linear dependency and it is useful to bias DIIS in favor of the SCF cycle that had the lowest energy using the factor `DIISBfac`. This is achieved by multiplying all diagonal elements of the DIIS matrix with this factor unless it is the Fock matrix/density which leads to the lowest energy. The default value for `DIISBfac` is 1.05.

The value of `DIISMaxEq` is the maximum number of old Fock matrices to remember. Values of 5-7 have been recommended, while other users store 10-15 Fock matrices. Should the standard DIIS not achieve convergence, some experimentation with this parameter can be worthwhile. In cases where DIIS causes problems in the beginning of the SCF, it may have to be invoked at a later stage. The start of the DIIS procedure is controlled by `DIISStart`. It has a default value of 0.2 Eh, which usually starts DIIS after 0-3 cycles. A different way of controlling the DIIS start is adjusting the value `DIISMaxIt`, which sets the maximum number of cycles after which DIIS will be started irrespective of the error value.

```
%scf
  # control of the DIIS procedure
  CNVDIIS     true  # default: true
  DIISStart   0.1   # default: 0.2
  DIISMaxIt   5     # default: 12
  DIISMaxEq   7     # default: 5
  DIISBFac    1.2   # default: 1.05
  DIISMaxC    15.0  # default: 10.0
  # more convenient:
  DIIS Start 0.1  MaxIt 5  MaxEq 7  BFac 1.2  MaxC 15.0  end
end
```

Note that for troublesome or lacking SCF convergence the `TRAH` algorithm should be used (see Sec. *Trust-Region Augmented Hessian (TRAH) SCF*). If not turned off explicitly, `TRAH` is switched on automatically whenever convergence problems are present by means of the `AutoTRAH` feature (see Sec. *Trust-Region Augmented Hessian (TRAH) SCF*).

2.6.5 Kolmar's DIIS (KDIIS)

An alternative algorithm that makes use of the DIIS concept is called KDIIS (Kolmar's DIIS[6, 7]) in ORCA. The KDIIS algorithm is designed to bring the orbital gradient of any energy expression to zero using a combination of DIIS extrapolation and first order perturbation theory. Thus, the method is diagonalization-free. In our hands it is superior to the standard DIIS algorithm in many cases, but not always. The algorithm is invoked with the keyword `! KDIIS` and is available for RHF, UHF and CASSCF.

2.6.6 Approximate Second Order SCF (SOSCF)

SOSCF is an approximately quadratically convergent variant of the SCF procedure [8, 9]. The theory is relatively involved and will not be described here. In short – SOSCF computes an initial guess to the inverse orbital Hessian and then uses the BFGS formula in a recursive way to update orbital rotation angles. As information from a few iterations accumulates, the guess to the inverse orbital Hessian becomes better and better and the calculation reaches a regime where it converges superlinearly. As implemented, the procedure converges as well or slightly better than DIIS and takes a somewhat less time. However, it is also *a lot* less robust, so that DIIS is the method of choice for many problems (see also the description of the full second-order trust-region augmented Hessian (`TRAH`) procedure in the next section). On the other hand, SOSCF is useful when DIIS gets stuck at some error around ~ 0.001 or 0.0001. Such cases were the primary motive for the implementation of SOSCF into ORCA.

The drawback of SOSCF is the following: in the beginning of the SCF, the orbital gradient (the derivative of the total energy with respect to rotations that describe the mixing of occupied and virtual MOs) is large, so that one is far from the quadratic regime. In such cases, the procedure is not successful and may even wildly diverge. Therefore it is recommended to only invoke the SOSCF procedure in the very end of the SCF where DIIS may lead to "trailing" convergence. SOSCF is controlled by the variables `SOSCFStart` and `SOSCFMaxIt`. `SOSCFStart` is a threshold for the orbital gradient. When the orbital gradient, or equivalently the DIIS Error, fall below `SOSCFStart`, the SOSCF procedure is initiated. `SOSCFMaxIt` is the latest iteration to start the SOSCF even if the orbital gradient is still above `SOSCFStart`.

```
%scf
  # control of the SOSCF procedure
  CNVSOSCF    true  # default: false
  SOSCFStart  0.1   # default: 0.01
  SOSCFMaxIt  5     # default: 1000
  # more convenient:
  SOSCF Start 0.1  MaxIt 5  end
end
```

For many calculations on transition metal complexes, it is a good idea to be conservative in the startup criterion for SOSCF, it may diverge otherwise. A choice of 0.01 or lower is recommended.

2.6.7 Trust-Region Augmented Hessian (TRAH) SCF

The **Trust-Region Augmented Hessian (TRAH)** method[10, 11, 12, 13] is a robust SCF convergence algorithm that ensures reliable convergence to the electronic ground state by using information from the electronic Hessian. It is especially useful for difficult SCF cases where standard DIIS/SOSCF approaches fail, such as open-shell systems or molecules with complex electronic structures.

When to Use TRAH

The TRAH-SCF procedure can be:

- **Activated manually** using the `! TRAH` keyword.
- **Triggered automatically** (`AutoTRAH`) when standard SCF convergence is too slow or diverges.

```
%scf
  AutoTRAH true
end
```

AutoTRAH

The automatic mode (`AutoTRAH`) monitors the convergence behavior by performing a log-scale linear interpolation of the electronic gradient norm over several SCF iterations. After a minimum number of SCF iterations (`AutoTRAHIter`, default: 20), the algorithm analyzes the last `AutoTRAHNInter` (default: 10) values of the gradient norm. A linear fit is performed in $\log_{10}(||\mathbf{g}||)$ space, and the slope s of this interpolation is used to assess convergence. If the slope indicates a convergence rate slower than 10^{-s}, with $s <$ `AutoTRAHTol` (default: 1.125), standard SCF is stopped and TRAH-SCF is initiated using the current set of orbitals.

This mechanism allows ORCA to switch to the more robust TRAH-SCF method only when necessary, improving efficiency for difficult SCF cases while avoiding unnecessary overhead for easy ones.

To disable automatic activation:

```
! NOTRAH
```

or

```
%scf
  AutoTRAH false
end
```

Theoretical Basis

TRAH builds a quadratic model of the SCF energy as a function of orbital rotations $\mathbf{x}$:

$$E(\mathbf{x}) = E_0 + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}, \quad ||\mathbf{x}|| \leq h$$

Minimizing $E(\mathbf{x})$ under the trust region constraint leads to level-shifted Newton equations:

$$(\mathbf{H} - \mu\mathbf{I})\mathbf{x} = -\mathbf{g}$$

Instead of solving these directly, the algorithm solves the eigenvalue problem of the scaled augmented Hessian matrix:

$$\begin{pmatrix} 0 & \alpha\mathbf{g}^T \\ \alpha\mathbf{g} & \mathbf{H} \end{pmatrix} \begin{pmatrix} 1 \\ \tilde{\mathbf{x}} \end{pmatrix} = \mu \begin{pmatrix} 1 \\ \tilde{\mathbf{x}} \end{pmatrix}$$

The Davidson algorithm is used for iterative diagonalization until the residual norm of $\tilde{\mathbf{x}}$ is below `TolFacMicro` × `||g||`. The scaling factor α is adjusted via bisection in the interval $[\alpha_0, \alpha_1]$ to ensure the trust region constraint is satisfied.

Performance and Implementation Details

- **Parallelization**: TRAH is MPI-parallel and supports large molecules using AO-based Fock matrices.
- **Acceleration**: TRAH supports `RIJ`, `RIJONX`, `RIJK`, and `RIJCOSX`, and is compatible with solvation models like `C-PCM` and `SMD`.
- **Available Methods**: Implemented for `RHF`, `RKS`, `UHF`, and `UKS`. `ROHF` and `ROKS` are not yet supported.
- **Efficient Hessian Operations**: Similar to CP-SCF and TD-DFT, gradient and Hessian operations use AO-basis sigma vectors for efficiency.

Grid parameters to speed up these steps:

```
%method
  Z_GridXC   1      // Lebedev Grid
  Z_IntAccXC 3.467  // eps parameter of radial grid
  Z_GridX    1      // Lebedev Grid
  Z_IntAccX  3.067  // eps parameter of radial grid
end
```

Input Parameters

```
%scf

 # AutoTRAH parameter
 AutoTRAH       true      # enables by default
 AutoTRAHTol    1.125     # slop of gradient convergence
 AutoTRAHIter   20        # first iteration for which TRAH could be called
 AutoTRAHNInter 10        # number of iteration for monitoring convergence progress

 # TRAH parameter
 trah
   MaxRed          24     # maximum number of Davidson micro iterations
   NStart          2      # number of start vectors for Davidson (at least 2)
   TolFacMicro     0.1    # Scaling factor for Davidson convergence
                          #  threshold =  TolFacMicro * || G ||
   MinTolMicro     1.e-2  # minimum accuracy of micro iterations
   QuadRegionStart 1.e-4  # start Newton-Raphson if || G || < QuadRegionStart
   tradius         0.4    # initial trust radius
   AlphaMin        0.1    # lower bound of gradient scaling parameter
   AlphaMax        1000.  # upper bound of gradient scaling parameter
   Randomize       true   # add white noise to Davidson start vectors
   PseudoRand      false  # use pseudo random numbers for comparibility
   MaxNoise        1.e-2  # maximum random number magnitude
   OrbUpdate       Taylor # orbital update algorithm (TAYLOR or CAYLEY)
   InactiveMOs     Canonical # CANONICAL or NOTSET
   Precond         Diag   # DIAG, FULL, or NONE
   PreconMaxRed    250    # maximum dimension of FULL Hessian for preconditioning
 end

 TolG 1.e-6  # SCF convergence threshold for gradient norm

end
```

- `Precond = FULL` uses the exact Hessian for a subset of important MO pairs, improving convergence for small RHF/UHF systems but not necessarily for large molecules or DFT.
- For `FULL`, ensure RI and auxiliary basis (`/C`) are provided.

Practical Notes

- **Quadratic Convergence**: Near the solution, TRAH achieves quadratic convergence.
- **Guaranteed Convergence**: In exact arithmetic and absent numerical noise, TRAH will always converge. If not, increase `MAXITER` or tighten numerical thresholds (e.g., grid accuracy).
- **LibXC Compatibility**: Some functionals lack ORCA-native kernels (e.g., `PWPB95`). Use `LIBXC(...)` for TRAH support.

Example Case

In a high-spin Rh_{12}^{+} cluster, DIIS fails to converge while TRAH steadily reduces the gradient norm below 10^{-6}:

Fig. 2.1: TRAH-SCF gradient norm of a PBE/def2-TZVP calculation for a Rh_{12}^{+} cluster in high-spin configuration ($M_s = 36$). Structure from Ref. [14].

2.6.8 Finite Temperature SCF

A finite temperature can be used to apply a Fermi-like occupation number smearing over the orbitals of the system, which may sometimes help to get convergence of the SCF equations in near hopeless cases. Through the smearing, the electrons are distributed according to Fermi statistics among the available orbitals. The "chemical potential" is found through the condition that the total number of electrons remains correct. Gradients can be computed in the presence of occupation number smearing.

```
%scf SmearTemp 5000  # ``temperature'' in Kelvin
 end
```

Note

- Finite temperature SCF (fractional occupation numbers or FOD analysis, see sections *Fractional Occupation Numbers* and *Fractional Occupation Number Weighted Density (FOD)*, respectively) *cannot* be used together with `SOSCF` methods.

Fractional Occupation Numbers

Only a very basic implementation of fractional occupation numbers is presently provided. It is meant to deal with orbitally degenerate states in the UHF/UKS method. Mainly it was implemented to avoid symmetry breaking in DFT calculations on orbitally degenerate molecules and atoms. The program checks the orbital energies of the initial guess orbitals, finds degenerate sets and averages the occupation numbers among them. Currently the criterion for degenerate orbitals is 10^{-3} Eh. The fractional occupation number option is invoked by:

```
%scf
  FracOcc true
end
```

Clearly, the power of fractional occupation numbers goes far beyond what is presently implemented in the program and future releases will likely make more use of them. The program prints a warning whenever it uses fractional occupation numbers. The fractionally occupied orbitals should be checked to ensure they are actually the intended ones.

Note

- Using `GuessMode = CMatrix` will cause problems because there are no orbital energies for the initial guess orbitals. The program will then average over *all* orbitals — which makes no sense at all.

2.6.9 Tips and Tricks: Converging SCF Calculations

Despite all efforts you may still find molecules where SCF convergence is poor. These are almost invariably related to open-shell situations and the answer is almost always to provide "better" starting orbitals. Here is my standard strategy to deal with this (assuming a DFT calculation):

- Perform a small basis set (`SV`) calculation in using the `LSD` or `BP` functional and RI approximation with a cheap auxiliary basis set. Set `Convergence=Loose` and `MaxIter=200` or so. The key point is to use a large damping factor and damp until the DIIS comes into a domain of convergence. This is accomplished by `SlowConv` or even `VerySlowConv`. If you have an even more pathological case you may need to set `DampFac` larger and `DampErr` smaller than chosen by these defaults. This calculation is quite crude and may take many cycles to converge. It will however be rather quick in terms of wall clock time. If the DIIS gets stuck at some error 0.001 or so the `SOSCF` (or even better `TRAH`) could be put in operation from this point on.
- Use the orbitals of this calculation and `GuessMode=CMatrix` to start a calculation with the target basis set. In DFT we normally use a pure GGA functional (e.g. BP86). This calculation normally converges relatively smoothly.
- Use the target functional, grid etc. to get the final calculation converged. In many cases this should converge fairly well now.

Here are a few other things that can be tried:

- Try to start from the orbitals of a related closed-shell species. In general closed-shell MO calculations tend to converge better. You then hope to reach the convergence radius of another converger for the open-shell case.
- Try to start from the orbitals of a more positive cation. Cation calculations tend to converge better.
- Try to start from a calculation with a smaller basis set. Smaller basis sets converge better. Then you have the choice of `GuessMode=CMatrix` or `GuessMode=FMatrix` which will affect the convergence behavior.
- Use large level shifts. This increases the number of iterations but stabilizes the converger. (`shift shift 0.5 erroff 0 end`)
- If you are doing DFT calculations try to start from a Hartree-Fock solution for your molecule. HF calculations tend to converge somewhat better because they have a larger HOMO-LUMO gap (there are of course exceptions).

- Carefully look at the starting orbitals (`Print[P_GuessOrb]=1`) and see if they make sense for your molecule. Perhaps you have to reorder them (using `Rotate`) to obtain smooth convergence.
- Most of the time the convergence problems come from "unreasonable" structures. Did you make sure that your coordinates are in the correct units (Angström or Bohrs?) and have been correctly recognized as such by the program?
- If you have trouble with UHF calculations try ROHF (especially SAHF or CAHF) first and then go to the UHF calculation.
- Fool around with `Guess=Hueckel`, `PAtom` or even `HCore`.
- It may sometimes be better to converge to an undesired state and then take the orbitals of this state, reorder them (using `Rotate`) and try to converge to the desired state.
- Similarly, bad orbitals may be manipulated using the SCF stability analysis (section *SCF Stability Analysis*) to provide a new guess.
- Try to start the calculation with a large damping factor (`DampFac=0.90`; or even larger) and specify a relatively small DIIS error to turn damping off (say `DampErr=0.02`;). This will increase the number of cycles but may guide you into a regime were the calculation actually converges.
- The advices above mostly apply to Hartree-Fock and DFT. For CASSCF, the available options and how they can aid to overcome convergence problems are described in the CASSCF manual section. In many cases modifying the initial guess or adding a level shift will help. Do not hesitate to use large level-shifts (e.g 2.0 or even 3.0). The manual is accompanied by CASSCF tutorial that goes through many details of the process including practical advices on convergence. The choice of initial guess is crucial. Some guesses work better for organic molecules while others excel for transition-metal complexes. The tutorial therefore discusses various initial guess options available in ORCA.
- If you managed to converge your system in another quantum chemistry program, and do not want to waste time re-converging it in ORCA, then you may convert the converged orbitals in the other program to the ORCA format using third-party tools like MOKIT (section *orca_2mkl: Old Molekel as well as Molden inputs*). However, keep in mind that converging a wavefunction in program A and reconverging it in program B comes with an inevitable overhead: program B usually has to perform a few extra SCF iterations even if the wavefunction is fully converged in program A, because (among many other reasons) the two programs in general use different integration grids. Therefore, use this method only when you expect a significant time saving.
- If nothing else helps, stop, grab a friend and go to the next pub (you can also send me an unfriendly e-mail but this will likely not make your calculation converge any quicker; ☺).

2.6.10 Local-SCF Method

The Local SCF (LSCF) method developed by X. Assfeld and J.-L. Rivail ([15]) allows optimising single determinant wave functions within the constraint of keeping some selected orbitals frozen (i.e. unchanged), whilst all orbitals fulfill orthogonality requirements. The LSCF method may be applied to unrestricted Hartree-Fock or DFT calculations.

The use of the LSCF method consists of selecting the set of orbitals to freeze during the SCF process according to the guess orbitals of the calculation. The selection is done by employing the `LSCFalpha` and/or `LSCFbeta` keywords in the SCF bloc for alpha and beta orbitals, respectively, followed by the orbital selection. **Intervals** are used to define the selection and one may go up to five intervals (5×2 numbers separate with commas). In the example below, `LSCFalpha 0,1,4,4,10,12` corresponds to freezing the alpha orbitals from 0 to 1 (**0,1**,4,4,10,12), orbital 4 (0,1,**4,4**,10,12), and from 10 to 12 (0,1,4,4,**10,12**). Accordingly, `LSCFbeta 0,3,5,5` corresponds to freezing beta orbitals 0 to 3 and orbital 5.

```
%scf
LSCFalpha 0,1,4,4,10,12
LSCFbeta  0,3,5,5
end
```

The energy of frozen orbitals is meaningless since they are not eigenfunctions of the Fockian. During the LSCF process, the energy of the occupied frozen orbitals is artificially defined as the lowest occupied orbital energy of the guess orbitals minus 1000 Hartree, while for the virtual frozen orbitals, the energy is defined as the highest virtual

orbital energy plus 1000 Hartree. Hence, all occupied frozen orbitals are sorted at the beginning of the orbital distribution and the virtual frozen orbitals at the end.

2.6.11 Keywords

Simple input keywords related to the SCF procedure are summarized in Section 2.6.11.

Table 2.11: Simple input keywords for the SCF procedure

Keyword	Description
Convergence acceleration	
`DIIS`	Enable *DIIS*
`NoDIIS`	Disable *DIIS*
`KDIIS`	Enable *Kollmar's DIIS*
`TRAH`	Enable *TRAH*
`NoTRAH`	Disable *TRAH*
`SOSCF`	Enable *SOSCF*
`NoSOSCF`	Disable *SOSCF*
`Damp`	Enable *damping*
`NoDamp`	Disable *damping*
`LShift`	Enable *level shifting*
`NoLShift`	Disable *level shifting*
Solver parameter selection (see Section 2.6.2)	
`EasyConv`	Assumes no convergence problems.
`NormalConv`	Default solver parameters
`SlowConv`	Selects appropriate SCF solver parameters for difficult cases. Most transition metal complexes fall into this category.
`VerySlowConv`	Selects appropriate SCF solver parameters for very difficult cases.
`SCFCONV<n>`	Adjusts `TolE` to 10e-n and most of the other tolerances according to Table 2.9 and Table 2.10 (not recommended!)
Fractional occupation numbers	
`FracOcc`	Enable fractional occupations (also compute *FOD*)
`Smear`	Enable occupation number smearing with `SmearTemp=5000` (also compute *FOD*)
`NoSmear`	Disable occupation number smearing
`FOD`	Perform *FOD analysis* with default settings (TPSS/def2-TZVP, TightSCF, SmearTemp = 5000 K)

2.7 Basis Sets

ORCA provides a large number of natively implemented *orbital* and *auxiliary basis sets* alongside various *effective core potentials (ECPs)* that can be combined with them. For use with *scalar-relativistic methods like ZORA, DKH, or X2C*, specialized *relativistic basis sets* are available as well. Furthermore, other basis sets can be *read from external files*. Most built-in basis sets and ECPs were obtained from the Basis Set Exchange[16] or its predecessor, the EMSL library, and the input format in ORCA is closely related to the "GAMESS-US" format.

2.7.1 Basic Usage

The easiest way to use orbital and auxiliary basis sets in ORCA is via the simple input keywords. All available orbital basis set keywords can be found in Section 2.7.2 and all auxiliary basis set options in Section 2.7.4. For example, the Karlsruhe def2-TZVP basis set can be invoked via the `def2-TZVP` keyword.

```
! def2-TZVP
```

Note

Some basis sets like the *Karlsruhe def2* employ *ECPs* by default for heavy elements, in this case the `def2-ECP`. The explicit control of the ECPs is described in Section 2.7.5.

Auxiliary basis sets needed for *resolution-of-the-identity (RI)* can be defined in the same way. In this example, Weigend's universal `def2/J` auxiliary basis is used.

```
! def2-TZVP def2/J
```

Note

Note that for many calculations, RI is activated by default (e.g. *RIJCOSX* for *hybrid DFT*). Accordingly, the `def2/J` auxiliary basis is automatically invoked by default as well if not otherwise specified. In scalar relativistic calculations, the default is `SARC/J` instead.

Note that there are three separate slots for auxiliary basis sets for RI:

- `AuxJ` is the Coulomb-fitting basis for the *RI-J*, *RIJDX/RIJONX*, and *RIJCOSX* approximations.
- `AuxJK` is the Coulomb- and exchange-fitting basis used for *RI-JK*.
- `AuxC` is used for RI-based integral generation steps in post-SCF dynamical electron correlation methods, such as *RI-MP2*, *DLPNO-MP2*, and *DLPNO-CC*.

Finally, *F12 methods* require a complementary auxiliary basis set (`CABS`), in addition to the *specialized orbital basis* (and possibly `AuxC`), for example:

```
! F12-RI-MP2 cc-pVDZ-F12 cc-pVDZ-F12-CABS cc-pVTZ-F12-MP2Fit
```

Specifying an auxiliary basis with the simple input keyword, assigns it to the corresponding slot. However, each basis slot, as well as the ECP, can be assigned explicitly in the `%basis` block. For example, a "/JK" basis may be assigned to `AuxJ` in this way.

```
%basis
  Basis "def2-TZVP"         # The orbital expansion basis set
  ECP   "def2-ECP"          # Effective core potential
  AuxJ  "def2/J"            # RI-J auxiliary basis set
  AuxJK "def2/JK"           # RI-JK auxiliary basis set
  AuxC  "def2-TZVP/C"       # Auxiliary basis set for correlated
                            # calculations, e.g. RI-MP2
  CABS  "cc-pVDZ-F12-OptRI" # complementary auxiliary basis set
                            # for F12 calculations
end
```

If required, all basis sets can be decontracted via simple input (e.g. `! DECONTRACT`) or the `%basis` block with the respective keywords below. Note that if your basis set arises from general contraction, it will contain duplicate primitives in several contractions. These will be removed such that only unique primitives remain and there is no problem with redundancy.

```
%basis
  DecontractBas   false  # if chosen "true" the program will
                         # decontract the orbital basis set
  DecontractAuxJ  false  # if "true" - decontract the AuxJ basis set
  DecontractAuxJK false  # if "true" - decontract the AuxJK basis set
  DecontractAuxC  false  # if "true" - decontract the AuxC basis set
  DecontractCABS  true   # if "false" - do not decontract the CABS
  Decontract      false  # if "true" - decontract all basis sets
end
```

Tip

- Generally, basis sets can be *assigned to specific elements*, to individual *atoms*, or even to *structural fragments*.
- Any built-in basis can be exported via the *`orca_exportbasis` utility*.
- References for the built-in basis sets are usually printed at the start of the ORCA output, as well as by `orca_exportbasis`.

Warning

ORCA uses pure d and f functions (5D and 7F instead of Cartesian 6D and 10F) for all basis sets. This needs to be taken into account when results are compared with other programs, especially for Pople-style basis sets that were optimized with Cartesian (6D) functions.

2.7.2 Orbital Basis Sets

In the following, we will give an outline of natively implemented basis sets from various families, like the *Karlsruhe def2* and *correlation consistent* basis sets, and list which elements are covered by each. If an *ECP* is necessary for heavy elements, it is documented in the respective table entries and invoked by default if the basis is selected in the simple input or via the `basis` keyword in the `%basis` block.

Pople Basis Sets

Various basis sets of the Pople basis set family are available in ORCA. A list of all available Pople-style basis sets is given in Table 2.12.

Naming Convention of Pople Basis Sets

- * or (d) adds one set of first polarization functions on all atoms except H
- ** or (d,p) adds one set of first polarization functions on all atoms
- Further combinations: (2d), (2df), (2d,p), (2d,2p), (2df,2p), (2df,2pd)
- + before "G" includes diffuse functions on all atoms except H (e.g. 6-31+G)
- ++ before "G" includes diffuse functions on all atoms. Works only when H polarization is already included, e.g. 6-31++G(d,p)

Table 2.12: Available Pople-style basis sets.

Basis Set	Elem.	ECP	Comment
`STO-3G`	H–I	–	Minimal basis set
`3-21G`	H–Cs	–	
`3-21GSP`	H–Ar	–	
`4-22GSP`	H–Ar	–	
`6-31G`	H–Zn	–	
`6-31G*`	H–Kr	–	
`m6-31G`	Sc–Cu	–	Modified 6-31G for 3d transition metals (Sc–Cu)
`m6-31G*`	Sc–Cu	–	
`6-31G**`	H–Zn	–	
`6-31G(d)`	H–Zn	–	
`6-31G(d,p)`	H–Zn	–	
`6-31G(2d)`	H–Zn	–	
`6-31G(2d,p)`	H–Zn	–	
`6-31G(2d,2p)`	H–Zn	–	
`6-31G(2df)`	H–Zn	–	
`6-31G(2df,2p)`	H–Zn	–	
`6-31G(2df,2pd)`	H–Zn	–	
`6-31+G*`	H–Kr	–	
`6-31+G**`	H–Zn	–	
`6-31+G(d)`	H–Zn	–	
`6-31+G(d,p)`	H–Zn	–	
`6-31+G(2d)`	H–Zn	–	
`6-31+G(2d,p)`	H–Zn	–	
`6-31+G(2d,2p)`	H–Zn	–	
`6-31+G(2df)`	H–Zn	–	
`6-31+G(2df,2p)`	H–Zn	–	
`6-31+G(2df,2pd)`	H–Zn	–	

continues on next page

Table 2.12 – continued from previous page

Basis Set	Elem.	ECP	Comment
6-31++G**	H–Zn	–	
6-31++G(d,p)	H–Zn	–	
6-31++G(2d,p)	H–Zn	–	
6-31++G(2d,2p)	H–Zn	–	
6-31++G(2df,2p)	H–Zn	–	
6-31++G(2df,2pd)	H–Zn	–	
6-311G	H–Br	–	
6-311G*	H–Br	–	
6-311G**	H–Br	–	
6-311G(d)	H–Br	–	
6-311G(d,p)	H–Br	–	
6-311G(2d)	H–Br	–	
6-311G(2d,p)	H–Br	–	
6-311G(2d,2p)	H–Br	–	
6-311G(2df)	H–Br	–	
6-311G(2df,2p)	H–Br	–	
6-311G(2df,2pd)	H–Br	–	
6-311G(3df)	H–Br	–	
6-311G(3df,3pd)	H–Br	–	
6-311+G*	H–Br	–	
6-311+G**	H–Br	–	
6-311+G(d)	H–Br	–	
6-311+G(d,p)	H–Br	–	
6-311+G(2d)	H–Br	–	
6-311+G(2d,p)	H–Br	–	
6-311+G(2d,2p)	H–Br	–	
6-311+G(2df)	H–Br	–	
6-311+G(2df,2p)	H–Br	–	
6-311+G(2df,2pd)	H–Br	–	
6-311+G(3df)	H–Br	–	
6-311+G(3df,2p)	H–Br	–	
6-311+G(3df,3pd)	H–Br	–	
6-311++G**	H–Br	–	
6-311++G(d,p)	H–Br	–	
6-311++G(2d,p)	H–Br	–	
6-311++G(2d,2p)	H–Br	–	
6-311++G(2df,2p)	H–Br	–	
6-311++G(2df, 2pd)	H–Br	–	
6-311++G(3df, 3pd)	H–Br	–	

Ahlrichs Basis Sets

The older Ahlrichs basis sets implemented in ORCA cover all-electron basis sets and the basis sets automatically employing the def-ECP for all elements beyond Rb. A list of available Ahlrichs basis sets is given in Table 2.13. Relativistically recontracted variants are shown in Section 2.7.3.

Note

Past versions of ORCA (ORCA <4.0) used to load all-electron basis sets also for elements Rb-I with the below keywords for double- and triple-ζ basis sets. The Rb-I basis sets originated from non-relativistic all-electron basis sets of the Turbomole library (such as "TZVPAlls"). **This automatic substitution is now deprecated**. However, we offer temporarily the ability to reproduce that behavior by adding the prefix "old-" to the below keywords, e.g. `old-TZVP`.

Table 2.13: Available all-electron Ahrlichs basis sets.

Basis Set	Elem.	ECP	Comment
SV	H–Kr	–	Valence double-zeta basis set.
SV(P)	H–Kr	–	Valence double-zeta with polarization only on heavy elements.
SVP	H–Kr	–	Polarized valence double-zeta basis set.
TZV	H–Kr	–	Valence triple-zeta basis set.
TZV(P)	H–Kr	–	Valence triple-zeta with polarization on heavy elements.
TZVP	H–Kr	–	Polarized valence triple-zeta basis set.
TZVPP	H–Kr	–	Doubly polarized triple-zeta basis set.
QZVP	H–Kr	–	Polarized valence quadruple-zeta basis set.
QZVPP	H–Kr	–	Doubly polarized quadruple-zeta basis set.
def-SV(P)	H–Lr	def-ECP (Rb–Lr)	Valence double-zeta with polarization only on heavy elements.
def-SVP	H–Lr	def-ECP (Rb–Lr)	Polarized valence double-zeta basis set.
def-TZVP	H–Lr	def-ECP (Rb–Lr)	Valence triple-zeta basis set with polarization functions.
def-TZVPP	H–Lr	def-ECP (Rb–Lr)	Doubly polarized triple-zeta basis set.
Minimally augmented (scheme by Truhlar et al.[17])			
ma-def-TZVP	Fr–Lr	def-ECP (Fr–Lr)	Minimally augmented def-TZVP basis set.
Legacy definitions (not recommended!)			
old-SV	H–I	–	
old-SV(P)	H–I	–	
old-SVP	H–I	–	
old-TZV	H–I	–	
old-TZV(P)	H–I	–	
old-TZVP	H–I	–	
old-TZVPP	H–I	–	

Karlsruhe def2 Basis Sets

The valence polarized basis sets of the Karlsruhe def2 family are all-electron for elements **H-Kr**, and automatically load Stuttgart-Dresden *effective core potentials* for elements **Rb-Rn**. A list of available Karlsruhe def2 basis sets is given in Table 2.14. Relativistically recontracted variants are discussed in Section 2.7.3.

Table 2.14: Available Karlsruhe def2 basis sets.

Basis Set	Elem.	ECP	Comment
def2-SVP	H–Rn	def2-ECP (Rb–Rn)	Polarized valence double-ζ.
def2-SV(P)	H–Rn	def2-ECP (Rb–Rn)	def2-SVP with slightly reduced polarization.
def2-TZVP	H–Rn	def2-ECP (Rb–Rn)	Polarized valence triple-ζ. Quite similar to the older def-TZVPP for main group elements and TZVP for H.
def2-TZVP(-f)	H–Rn	def2-ECP (Rb–Rn)	def2-TZVP with f polarization removed from main group elements.
def2-TZVPP	H–Rn	def2-ECP (Rb–Rn)	Doubly polarized valence triple-ζ.
def2-QZVP	H–Rn	def2-ECP (Rb–Rn)	Polarized valence quadruple-ζ.
def2-QZVPP	H–Rn	def2-ECP (Rb–Rn)	Doubly polarized valence quadruple-ζ.
Diffuse (Rappoport et al.[18, 19])			
def2-SVPD	H–Rn	def2-ECP (Rb–Rn)	Diffuse def2-SVP basis set for property calculations
def2-TZVPD	H–Rn	def2-ECP (Rb–Rn)	Diffuse def2-TZVP basis set for property calculations
def2-TZVPPD	H–Rn	def2-ECP (Rb–Rn)	Diffuse def2-TZVPP basis set for property calculations
def2-QZVPD	H–Rn	def2-ECP (Rb–Rn)	Diffuse def2-QZVP basis set for property calculations
def2-QZVPPD	H–Rn	def2-ECP (Rb–Rn)	Diffuse def2-QZVPP basis set for property calculations
Minimally augmented (scheme by Truhlar et al.[17])			
ma-def2-SVP	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-SVP basis set.
ma-def2-SV(P)	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-SV(P) basis set.
ma-def2-mSVP	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-mSVP basis set.
ma-def2-TZVP	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-TZVP basis set.
ma-def2-TZVP(-f)	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-TZVP(-f) basis set.
ma-def2-TZVPP	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-TZVPP basis set.
ma-def2-QZVPP	H–Rn	def2-ECP (Rb–Rn)	Minimally augmented def2-QZVPP basis set.

Karlsruhe dhf Basis Sets

These basis sets are derived from the def2-XVP ones with small modifications for 5s, 6s, 4d, and 5d elements and iodine.[20] They are optimized for the revised Dirac-Fock ECPs (`dhf-ECP`) as opposed to the Wood–Boring ones (`def2-ECP`). For elements H–Kr these basis sets are equivalent to the respective *def2-XVP* basis set. Versions for two-component methods are also available, e.g. `dhf-TZVP-2c`, **however, such methods are currently not implemented in ORCA**. A list of available Karlsruhe dhf basis sets is given in Table 2.15.

Table 2.15: Available dhf basis sets.

Basis Set	Elem.	ECP	Comment
`dhf-SV(P)`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-SV(P)
`dhf-SVP`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-SVP
`dhf-TZVP`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-TZVP
`dhf-TZVPP`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-TZVPP
`dhf-QZVP`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-QZVP
`dhf-QZVPP`	H–Rn	`dhf-ECP` (Rb–Rn)	based on def2-QZVPP
Two-component variants:			
`dhf-SVP-2c`	H–Rn	`dhf-ECP-2c` (Rb–Rn)	based on def2-SVP
`dhf-TZVP-2c`	H–Rn	`dhf-ECP-2c` (Rb–Rn)	based on def2-TZVP
`dhf-TZVPP-2c`	H–Rn	`dhf-ECP-2c` (Rb–Rn)	based on def2-TZVPP
`dhf-QZVP-2c`	H–Rn	`dhf-ECP-2c` (Rb–Rn)	based on def2-QZVP
`dhf-QZVPP-2c`	H–Rn	`dhf-ECP-2c` (Rb–Rn)	based on def2-QZVPP

Jensen Basis Sets

Jensen's polarization-consistent generally contracted basis sets include various general purpose and sepcialized variants. A list of available Jensen basis sets is given in Table 2.16.[21, 22, 23, 24, 25, 26, 27]

Table 2.16: Available Jensen basis sets.

Basis Set	Elem.	ECP	Comment
`pc-0`	H–Ca, Ga–Kr	–	
`pc-1`	H–Kr	–	
`pc-2`	H–Kr	–	
`pc-3`	H–Kr	–	
`pc-4`	H–Kr	–	
`aug-pc-0`	H–Ca, Ga–Kr	–	`pc-0` augmented by diffuse functions
`aug-pc-1`	H–Kr	–	`pc-1` augmented by diffuse functions
`aug-pc-2`	H–Kr	–	`pc-2` augmented by diffuse functions
`aug-pc-3`	H–Kr	–	`pc-3` augmented by diffuse functions
`aug-pc-4`	H–Kr	–	`pc-4` augmented by diffuse functions
Segmented contraction variants:			
`pcseg-0`	H–Kr	–	
`pcseg-1`	H–Kr	–	
`pcseg-2`	H–Kr	–	
`pcseg-3`	H–Kr	–	
`pcseg-4`	H–Kr	–	
`aug-pcseg-0`	H–Kr	–	`pcseg-0` augmented by diffuse functions
`aug-pcseg-1`	H–Kr	–	`pcseg-1` augmented by diffuse functions
`aug-pcseg-2`	H–Kr	–	`pcseg-2` augmented by diffuse functions
`aug-pcseg-3`	H–Kr	–	`pcseg-3` augmented by diffuse functions
`aug-pcseg-4`	H–Kr	–	`pcseg-4` augmented by diffuse functions
Optimized for nuclear magnetic shieldings:			
`pcSseg-0`	H–Kr	–	
`pcSseg-1`	H–Kr	–	
`pcSseg-2`	H–Kr	–	
`pcSseg-3`	H–Kr	–	
`pcSseg-4`	H–Kr	–	
`aug-pcSseg-0`	H–Kr	–	`pcSseg-0` augmented by diffuse functions
`aug-pcSseg-1`	H–Kr	–	`pcSseg-1` augmented by diffuse functions
`aug-pcSseg-2`	H–Kr	–	`pcSseg-2` augmented by diffuse functions
`aug-pcSseg-3`	H–Kr	–	`pcSseg-3` augmented by diffuse functions
`aug-pcSseg-4`	H–Kr	–	`pcSseg-4` augmented by diffuse functions

continues on next page

Table 2.16 – continued from previous page

Basis Set	Elem.	ECP	Comment
Optimized for spin-spin coupling constants:			
pcJ-0	H–He, B–Ne, Al–Ar	–	
pcJ-1	H–He, B–Ne, Al–Ar	–	
pcJ-2	H–He, B–Ne, Al–Ar	–	
pcJ-3	H–He, B–Ne, Al–Ar	–	
pcJ-4	H–He, B–Ne, Al–Ar	–	
aug-pcJ-0	H–He, B–Ne, Al–Ar	–	pcJ-0 augmented by diffuse functions
aug-pcJ-1	H–He, B–Ne, Al–Ar	–	pcJ-1 augmented by diffuse functions
aug-pcJ-2	H–He, B–Ne, Al–Ar	–	pcJ-2 augmented by diffuse functions
aug-pcJ-3	H–He, B–Ne, Al–Ar	–	pcJ-3 augmented by diffuse functions
aug-pcJ-4	H–He, B–Ne, Al–Ar	–	pcJ-4 augmented by diffuse functions
Optimized for hyperfine coupling constants:			
pcH-1	H, He, B–Ne, Al–Ar	–	
pcH-2	H, He, B–Ne, Al–Ar	–	
pcH-3	H, He, B–Ne, Al–Ar	–	
pcH-4	H, He, B–Ne, Al–Ar	–	
aug-pcH-1	H, He, B–Ne, Al–Ar	–	pcH-1 augmented by diffuse functions
aug-pcH-2	H, He, B–Ne, Al–Ar	–	pcH-2 augmented by diffuse functions
aug-pcH-3	H, He, B–Ne, Al–Ar	–	pcH-3 augmented by diffuse functions
aug-pcH-4	H, He, B–Ne, Al–Ar	–	pcH-4 augmented by diffuse functions
Optimized for core-spectroscopy:			
pcX-1	Li–Ar	–	
pcX-2	Li–Ar	–	
pcX-3	Li–Ar	–	
pcX-4	Li–Ar	–	
aug-pcX-1	Li–Ar	–	pcX-1 augmented by diffuse functions
aug-pcX-2	Li–Ar	–	pcX-2 augmented by diffuse functions
aug-pcX-3	Li–Ar	–	pcX-3 augmented by diffuse functions
aug-pcX-4	Li–Ar	–	pcX-4 augmented by diffuse functions

Hydrogenic Gaussian Basis Sets

Lehtolas hydrogenic Gaussian basis sets (HGBS) were constructed as widely transferable and accurate basis sets based on one-electron model systems imitating real atoms and molecules.[28] A list of available HGBS basis sets is given in Table 2.17.

Naming convention of HGBS basis sets

- m: represents the energy optimization threshold 10^{-m}
- Pn: represents the number of polarization shells
- A: indicates explicit augmentation by diffuse functions

Table 2.17: Karlsruhe HGBS, HGBSP, AHGBS, and AHGBSP basis sets.

Basis Set	Elem.	ECP	Comment
HGBS-5	H–Og	–	
HGBS-7	H–Og	–	
HGBS-9	H–Og	–	
HGBSP1-5	H–Og	–	Polarized variant
HGBSP1-7	H–Og	–	
HGBSP1-9	H–Og	–	
HGBSP2-5	H–Og	–	
HGBSP2-7	H–Og	–	
HGBSP2-9	H–Og	–	
HGBSP3-5	H–Og	–	
HGBSP3-7	H–Og	–	
HGBSP3-9	H–Og	–	
AHGBS-5	H–Og	–	Augmented by diffuse functions
AHGBS-7	H–Og	–	
AHGBS-9	H–Og	–	
AHGBSP1-5	H–Og	–	Polarized variant augmented by diffuse functions
AHGBSP1-7	H–Og	–	
AHGBSP1-9	H–Og	–	
AHGBSP2-5	H–Og	–	
AHGBSP2-7	H–Og	–	
AHGBSP2-9	H–Og	–	
AHGBSP3-5	H–Og	–	
AHGBSP3-7	H–Og	–	
AHGBSP3-9	H–Og	–	

Sapporo Basis Sets

The Sapporo basis set family, named after the city of Sapporo, Japan, comprises segmented contracted all-electron Gaussian basis sets up to quadruple-ζ quality.[29, 30, 31] They were developed as compact yet accurate basis sets focusing on calculations with correlated wavefunction methods. Variants optimized for the scalar-relativistic DKH3 Hamiltonian are also available. A list of available Sapporo basis sets is given in Table 2.18. Relativistic variants are discussed in Section 2.7.3.

Table 2.18: List of available Sapporo basis sets.

Basis Set	Elem.	ECP	Comment
Sapporo-DZP-2012	H–Xe	–	
Sapporo-TZP-2012	H–Xe	–	
Sapporo-QZP-2012	H–Xe	–	

Partridge Basis Sets

The basis sets of the Partridge family are uncontracted RHF groundstate energy-optimized.[32, 33] A list of available Partridge basis sets is given in Table 2.19.

Table 2.19: AvailablePartridge basis sets.

Basis Set	Elem.	ECP	Comment
Partridge-1	H, Li–Sr	–	
Partridge-2	H, Li–Kr	–	
Partridge-3	H, Li–Zn	–	
Partridge-4	Sc–Zn	–	

CRENB Basis Sets

The Christiansen-Ross-Ermler-Nash-Bursten (CRENB) basis sets and *effective core potentials* are available for most elements.[34, 35, 36, 37, 38, 39, 40] Note that while the ECPs were originally designed for spin-orbit coupling (SOC) interactions, **SOC calculations with ECPs are currently not implemented in ORCA**. A list of availble CRENB basis sets is given in Table 2.20.

Table 2.20: List of available CRENB basis sets.

Basis Set	Elem.	ECP	Comment
`CRENBL`	H, Li–Og	`CRENBL-ECP` (Li–Og)	*Large* basis version for use with *small*-core ECP

LANL Basis Sets

The Los Alamos National Laboratory (LANL) basis sets and *effective core potentials* were originally introduced in 1985 by Hay and Wadt.[41, 42, 43] Later polarized variants were proposed[44, 45] and the basis sets were extended by Roy et al.[46]. A list of available LANL basis sets is given in Table 2.21.

Table 2.21: List of available LANL basis sets.

Basis Set	Elem.	ECP	Comment
`LANL08`	Na–La, Hf–Bi	`HayWadt` (Na–La, Hf–Bi)	uncontracted
`LANL08(f)`	Sc–Cu, Y–Ag, La, Hf–Au	`HayWadt` (Sc–Cu, Y–Ag, La, Hf–Au)	uncontracted + polarization
`LANL2DZ`	H, Li–La, Hf–Bi, U–Pu	`HayWadt` (Na–La, Hf–Bi, U–Pu)	double-ζ, D95V for H–Ne
`LANL2TZ`	Sc–Zn, Y–Cd, La, Hf–Hg	`HayWadt` (Sc–Zn, Y–Cd, La, Hf–Hg)	triple-ζ
`LANL2TZ(f)`	Sc–Cu, Y–Ag, La, Hf–Au	`HayWadt` (Sc–Cu, Y–Ag, La, Hf–Au)	triple-ζ + polarization

Correlation-consistent Basis Sets

The correlation-consistent basis sets were pioneered by Dunning and co-workers and manifold variants have been developed since. They are among the most widely used for application with correlated wave-function methods and are particularly suited for *basis set extrapolation*. A list of available correlation-consisten basis sets is given in Table 2.22. Relativistic variants are discussed in Section 2.7.3.

Table 2.22: List of available correlation-consistent basis sets.

Basis Set	Elem.	ECP	Comment
`cc-pVDZ`	H–Ar, Ca–Kr	–	Dunning correlation-consistent polarized double-ζ
`cc-pVTZ`	H–Ar, Ca–Kr, Y, Ag, Au	–	Dunning correlation-consistent polarized triple-ζ
`cc-pVQZ`	H–Ar, Ca–Kr	–	Dunning correlation-consistent polarized quadruple-ζ
`cc-pV5Z`	H–Ar, Ca–Kr	–	Dunning correlation-consistent polarized quintuple-ζ
`cc-pV6Z`	H–He, Be–Ne, Al–Ar	–	Dunning correlation-consistent polarized sextuple-ζ
`aug-cc-pVDZ`	H–Ar, Sc–Kr	–	`cc-pVDZ` augmented by diffuse functions
`aug-cc-pVTZ`	H–Ar, Sc–Kr, Ag, Au	–	`cc-pVTZ` augmented by diffuse functions
`aug-cc-pVQZ`	H–Ar, Sc–Kr	–	`cc-pVQZ` augmented by diffuse functions
`aug-cc-pV5Z`	H–Ar, Sc–Kr	–	`cc-pV5Z` augmented by diffuse functions
`aug-cc-pV6Z`	H–He, B–Ne, Al–Ar	–	`cc-pV6Z` augmented by diffuse functions
With tight d functions:			
`cc-pVD(+d)Z`	Na–Ar	–	
`cc-pVT(+d)Z`	Na–Ar	–	
`cc-pVQ(+d)Z`	Na–Ar	–	
`cc-pV5(+d)Z`	Na–Ar	–	
`aug-cc-pVD(+d)Z`	Al–Ar	–	`cc-pVD(+d)Z` augmented by diffuse functions

continues on next page

Table 2.22 – continued from previous page

Basis Set	Elem.	ECP	Comment
aug-cc-pVT(+d)Z	Al–Ar	–	cc-pVT(+d)Z augmented by diffuse functions
aug-cc-pVQ(+d)Z	Al–Ar	–	cc-pVQ(+d)Z augmented by diffuse functions
aug-cc-pV5(+d)Z	Al–Ar	–	cc-pV5(+d)Z augmented by diffuse functions
aug-cc-pV6(+d)Z	Al–Ar	–	cc-pV6(+d)Z augmented by diffuse functions
Partially augmented according to Truhlar et al.[47]:			
apr-cc-pV(Q+d)Z	H–Ar	–	Augmented with sp diffuse functions on Li–Ca
may-cc-pV(T+d)Z	H–Ar	–	Augmented with sp diffuse functions on Li–Ca
may-cc-pV(Q+d)Z	H–Ar	–	Augmented with spd diffuse functions on Li–Ca
jun-cc-pV(D+d)Z	H–Ar	–	Augmented with sp diffuse functions on Li–Ca
jun-cc-pV(T+d)Z	H–Ar	–	Augmented with spd diffuse functions on Li–Ca
jun-cc-pV(Q+d)Z	H–Ar	–	Augmented with spdf diffuse functions on Li–Ca
jul-cc-pV(D+d)Z	H–Ar	–	Augmented with spd diffuse functions on Li–Ca
jul-cc-pV(T+d)Z	H–Ar	–	Augmented with spdf diffuse functions on Li–Ca
jul-cc-pV(Q+d)Z	H–Ar	–	Augmented with spdfg diffuse functions on Li–Ca
maug-cc-pV(D+d)Z	H–Ar	–	Equals jun-cc-pV(D+d)Z
maug-cc-pV(T+d)Z	H–Ar	–	Equals may-cc-pV(T+d)Z
maug-cc-pV(Q+d)Z	H–Ar	–	Equals apr-cc-pV(Q+d)Z
Core-polarized for core-valence electron correlation:			
cc-pCVDZ	H–Ar, Ca, Ga–Kr	–	Equals cc-pVDZ for H and He
cc-pCVTZ	H–Ar, Ca, Ga–Kr	–	Equals cc-pVTZ for H and He
cc-pCVQZ	H–Ar, Ca, Ga–Kr	–	Equals cc-pVQZ for H and He
cc-pCV5Z	H–Ar, Ca, Ga–Kr	–	Equals cc-pV5Z for H and He
cc-pCV6Z	H–He, B–Ne, Al–Ar	–	Equals cc-pV6Z for H and He
aug-cc-pCVDZ	H–Ar, Ga–Kr	–	cc-pCVDZ augmented by diffuse functions, equals aug-cc-pVDZ for H and He
aug-cc-pCVTZ	H–Ar, Ga–Kr	–	cc-pCVTZ augmented by diffuse functions, equals aug-cc-pVTZ for H and He
aug-cc-pCVQZ	H–Ar, Ga–Kr	–	cc-pCVQZ augmented by diffuse functions, equals aug-cc-pVQZ for H and He
aug-cc-pCV5Z	H–Ar, Ga–Kr	–	cc-pCV5Z augmented by diffuse functions, equals aug-cc-pV5Z for H and He
aug-cc-pCV6Z	H–He, B–Ne, Al–Ar	–	cc-pCV6Z augmented by diffuse functions, equals aug-cc-pV6Z for H and He
Core-polarized with weighted core functions:			
cc-pwCVDZ	H–Ar, Ca, Ga–Kr	–	Equals cc-pVDZ for H and He
cc-pwCVTZ	H–Ar, Ca–Kr, Ag, Au	–	Equals cc-pVTZ for H and He
cc-pwCVQZ	H–Ar, Ca–Kr	–	Equals cc-pVQZ for H and He
cc-pwCV5Z	H–Ar, Ca–Kr	–	Equals cc-pV5Z for H and He
aug-cc-pwCVDZ	H–Ar, Ga–Kr	–	cc-pwCVDZ augmented by diffuse functions, equals aug-cc-pVDZ for H and He
aug-cc-pwCVTZ	H–Ar, Sc–Kr, Ag, Au	–	cc-pwCVTZ augmented by diffuse functions, equals aug-cc-pVTZ for H and He
aug-cc-pwCVQZ	H–Ar, Sc–Kr	–	cc-pwCVQZ augmented by diffuse functions, equals aug-cc-pVQZ for H and He
aug-cc-pwCV5Z	H–Ar, Sc–Kr	–	cc-pwCV5Z augmented by diffuse functions, equals aug-cc-pV5Z for H and He
Pseudo-potential (ECP) variants:			
cc-pVDZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pVTZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pVQZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pV5Z-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
aug-cc-pVDZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pVDZ-PP augmented by diffuse functions

continues on next page

Table 2.22 – continued from previous page

Basis Set	Elem.	ECP	Comment
aug-cc-pVTZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pVTZ-PP augmented by diffuse functions
aug-cc-pVQZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pVQZ-PP augmented by diffuse functions
aug-cc-pV5Z-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pV5Z-PP augmented by diffuse functions
cc-pCVDZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	
cc-pCVTZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	
cc-pCVQZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	
cc-pCV5Z-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	
aug-cc-pCVDZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	cc-pCVDZ-PP augmented by diffuse functions
aug-cc-pCVTZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	cc-pCVTZ-PP augmented by diffuse functions
aug-cc-pCVQZ-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	cc-pCVQZ-PP augmented by diffuse functions
aug-cc-pCV5Z-PP	Ca, Sr, Ba, Ra	SK-MCDHF-RSC (Ca, Sr, Ba, Ra)	cc-pCV5Z-PP augmented by diffuse functions
cc-pwCVDZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pwCVTZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pwCVQZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra, U	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
cc-pwCV5Z-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	
aug-cc-pwCVDZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pwCVDZ-PP augmented by diffuse functions
aug-cc-pwCVTZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pwCVTZ-PP augmented by diffuse functions
aug-cc-pwCVQZ-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pwCVQZ-PP augmented by diffuse functions
aug-cc-pwCV5Z-PP	Ca, Cu–Kr, Sr–Xe, Ba, Hf–Rn, Ra	SK-MCDHF-RSC (Ca, Cu–Kr, Sr–Xe, Ba, Hf–Ra, U)	cc-pwCV5Z-PP augmented by diffuse functions
Optimized for hyperfine coupling constants:			
aug-cc-pVTZ-J	H, B–F, Al–Cl, Sc–Zn, Se	–	Sauer's basis set for accurate hyperfine coupling
W4 theory:			
haV(T+d)Z	H–Ar	–	cc-pVTZ (H–Be, Na, Mg), aug-cc-pVTZ (B–Ne), aug-cc-pVT(+d)Z (Al–Ar)
haV(Q+d)Z	H–Ar	–	cc-pVQZ (H–Be, Na, Mg), aug-cc-pVQZ (B–Ne), aug-cc-pVQ(+d)Z (Al–Ar)

continues on next page

Table 2.22 – continued from previous page

Basis Set	Elem.	ECP	Comment
haV(5+d)Z	H–Ar	–	cc-pV5Z (H–Be, Na, Mg), aug-cc-pV5Z (B–Ne), aug-cc-pV5(+d)Z (Al–Ar)

F12 Basis Sets

Special orbital basis sets for *F12 calculations* (larger than the regular D, T, Q-zeta basis sets!). A list of available F12 basis sets is given in Table 2.23. See Table 2.37 for the necessary complementary auxiliary basis sets (CABS).

Table 2.23: F12 basis sets.

Basis Set	Elem.	ECP	Comment
cc-pVDZ-F12	H–Ar	–	
cc-pVTZ-F12	H–Ar	–	
cc-pVQZ-F12	H–Ar	–	
Core-polarized:			
cc-pCVDZ-F12	Li–Ar	–	
cc-pCVTZ-F12	Li–Ar	–	
cc-pCVQZ-F12	Li–Ar	–	
Pseudo-potential (ECP) variants:			
cc-pVDZ-PP-F12	Ga–Kr, In–Xe, Tl–Rn	SK-MCDHF-RSC (Ga–Kr, In–Xe, Tl–Rn)	
cc-pVTZ-PP-F12	Ga–Kr, In–Xe, Tl–Rn	SK-MCDHF-RSC (Ga–Kr, In–Xe, Tl–Rn)	
cc-pVQZ-PP-F12	Ga–Kr, In–Xe, Tl–Rn	SK-MCDHF-RSC (Ga–Kr, In–Xe, Tl–Rn)	

Atomic Natural Orbital Basis Sets

Atomic natural orbitals are a special class of basis sets. They are represented by the orthonormal set of orbitals that diagonalizes a spherically symmetric, correlated atomic density. The idea is to put as much information as possible into each basis functions such that one obtains the best possible result with the given number of basis functions. This is particularly important for correlated calculations where the number of primitives is less an issue than the number of basis functions. ORCA features some ANO basis sets on the basis of the cc-pV6Z (or pc-4 where missing) basis set primitives.[48] These are very accurate and significantly better than the cc-pVnZ counterparts for the same number of basis functions (but much larger number of primitives). A list of available ANO basis sets is given in Table 2.24.

Note

- **aug-ANO-pVnZ:** full augmentation with spd, spdf, spdfg set of polarization functions. Almost as expensive as the next higher basis set. In fact, aug-ANO-pVnZ = ANO-pV$(n+1)$Z with the highest angular momentum polarization function deleted.
- **saug-ANO-pVnZ:** augmentation with a single set of sp functions. Greatly enhances the accuracy of the SCF energies but not for correlation energies.

Table 2.24: Available atomic natural orbital (ANO) basis sets.

Basis Set	Elem.	ECP	Comment
ANO-SZ	H–Ar, Sc–Zn	–	
ANO-pVDZ	H–Ar, Sc–Zn	–	
ANO-pVTZ	H–Ar, Sc–Zn	–	
ANO-pVQZ	H–Ar, Sc–Zn	–	
ANO-pV5Z	H–Ar, Sc–Zn	–	
ANO-pV6Z	H–Ar, Sc–Zn	–	
aug-ANO-pVDZ	H–Ar, Sc–Zn	–	ANO-pVDZ augmented by diffuse functions
aug-ANO-pVTZ	H–Ar, Sc–Zn	–	ANO-pVTZ augmented by diffuse functions
aug-ANO-pVQZ	H–Ar, Sc–Zn	–	ANO-pVQZ augmented by diffuse functions
aug-ANO-pV5Z	H–Ar, Sc–Zn	–	ANO-pV5Z augmented by diffuse functions
saug-ANO-pVDZ	H–Ar, Sc–Zn	–	ANO-pVDZ augmented by by a single set of diffuse sp functions
saug-ANO-pVTZ	H–Ar, Sc–Zn	–	ANO-pVTZ augmented by by a single set of diffuse sp functions
saug-ANO-pVQZ	H–Ar, Sc–Zn	–	ANO-pVQZ augmented by by a single set of diffuse sp functions
saug-ANO-pV5Z	H–Ar, Sc–Zn	–	ANO-pV5Z augmented by by a single set of diffuse sp functions

Efficient Calculations with ANO Basis Sets

Usually, ANO basis sets are "generally contracted" which means that for any given angular momentum all primitives contribute to all basis functions. Since the concept of ANOs only makes sense if the underlying set of primitives is large, the calculations readily become very expensive unless special precaution is taken in the integral evaluation algorithms. ORCA features special algorithms for ANO basis sets together with accurate ANO basis sets for non-relativistic calculations. However, even then the integral evaluation is so expensive that efficiency can only be realized if all integrals are stored on disk and are re-used as needed.

Currently, the use of ANOs is restricted to the *built-in ANO basis sets*. These are built upon the cc-pV6Z primitives and hence, the calculations take significant time.

Hint

Geometry optimizations with ANOs are discouraged; they will be *very* inefficient.

The use of ANOs is recommended in the following way:

```
! ano-pVTZ Conv TightSCF CCSD(T)
%maxcore 2000
* int 0 1
C  0 0 0    0   0   0
O  1 0 0  1.2   0   0
H  1 2 0  1.1 120   0
H  1 2 3  1.1 120 180
*
```

This yields:

```
ano-pVTZ:
E(SCF) = -113.920388785
E(corr)=   -0.427730189
```

Compare to the cc-pVTZ value of:

```
cc-pVTZ:
E(SCF) = -113.911870901
E(corr)=   -0.421354947
```

Thus, the ANO-based SCF energy is ca. 8–9 mEh lower and the correlation energy almost 2 mEh lower than with the cc-basis set of the same size. Usually, the ANO results are much closer to the basis set limit than the cc-results. Also, ANO values extrapolate very well (see section *Automatic extrapolation to the basis set limit*)

Importantly, the integrals are all stored in this job. Depending on your system and your patience, this may be possible up to 300–500 basis functions. The ORCA correlation modules have been rewritten such that they deal efficiently with these stored integrals. Thus, we might as well have used `! MO-CCSD(T)` or `! AO-CCSD(T)` , both of which would perform well.

Yet, the burden of generating and storing all four-index integrals quickly becomes rather heavy. Hence, the combination of ANO basis sets with the RI-JK technique is particularly powerful and efficient. For example:

```
! ano-pVTZ cc-pVTZ/JK RI-JK Conv TightSCF RI-CCSD(T)
```

For the SCF, this works very well and allows for much larger ANO based calculations to be done efficiently. Also, RI-MP2 can be done very efficiently in this way. However, for higher order correlation methods such as CCSD(T) the logical choice would be RI-CCSD(T) which is distinctly less efficient than the AO or MO based CCSD(T) (roughly a factor of two slower). Hence, ORCA implements a hybrid method where the RI approximation is used to generate all four index integrals. This is done via the "RI-AO" keyword:

```
! ano-pVTZ cc-pVTZ/JK RI-AO Conv TightSCF AO-CCSD(T)
```

In this case either AO-CCSD(T) or MO-CCSD(T) would both work well. This does not solve the storage bottleneck with respect to the four index integrals of course. If this becomes a real issue, then RI-CCSD(T) is mandatory. The error in the total energy is less than 0.1 mEh in the present example.

Warning

With conventional RI calculations the use of a second fit basis set is not possible and inconsistent results will be obtained. Hence, stick to one auxiliary basis!

Miscellaneous and Specialized Basis Sets

A list of further available specialized and miscellaneous basis sets is given in Table 2.25.

Table 2.25: Collection of available miscellaneous and specialized basis sets.

Basis Set	Elem.	ECP	Comment
D95	H, Li, B–Ne, Al–Cl	–	Dunning's double-ζ basis set
D95p	H, Li, B–Ne, Al–Cl	–	Polarized version of D95
EPR-II	H, B–F	–	Barone's double-ζ basis set for EPR calculations
EPR-III	H, B–F	–	Barone's triple-ζ basis set for EPR calculations
IGLO-II	H, B–F, Al–Cl	–	Kutzelnigg's basis set for NMR and EPR calculations.
IGLO-III	H, B–F, Al–Cl	–	Kutzelnigg's larger basis set for NMR and EPR calculations.
UGBS	H–Th, Pu–Am, Cf–Lr	–	Universal Gaussian basis set.
CP	Sc–Zn	–	
CP(PPP)	Sc–Zn	–	
Wachters+f	Sc–Cu	–	
W1-mtsmall	H–Ar	–	W1 theory basis set
W1-DZ	H–Ar	–	W1 theory basis set
W1-TZ	H–Ar	–	W1 theory basis set
W1-QZ	H–Ar	–	W1 theory basis set
W1-Opt	H–Ar	–	W1 theory basis set
MINI	H–Ca	–	Huzinaga's minimal basis set.
MINIS	H–Ca	–	Scaled version of the MINI
MIDI	H–Na, Al–K	–	Huzinaga's valence double-ζ basis set
MINIX	H–Lr	def-ECP (Rb–Lr)	Combination of small basis sets by Grimme (see Table 3.30). Used in *HF-3c*.
def2-mSVP	H–Lr	def2-ECP (Rb–Rn), def-ECP (Fr–Lr)	Used in *PBEh-3c* and *B3LYP-3c*
def2-mTZVP	H–Lr	def2-ECP (Rb–Rn), def-ECP (Fr–Lr)	Used in *B97-3c*
def2-mTZVPP	H–Lr	def2-ECP (Rb–Rn), def-ECP (Fr–Lr)	Used in *r²SCAN-3c*
vDZP	H–Rn	vDZP-ECP (B–Rn)	Molecule-optimized polarized valence double-ζ basis set by Grimme et al. Used in *ωB97X-3c*.[49]

2.7.3 Relativistic Basis Sets

Scalar-relativistic calculations with the *DKH*, *ZORA* or *X2C* approaches typically require specifically optimized or recontracted basis sets – see *Basis Sets in Relativistic Calculations* for details. ORCA provides various suitable basis sets for such calculations like the recontracted *Karlsruhe*, the *SARC*, and *SARC2*, and various optimized all-electron *correlation-consistent* basis sets. Suitable auxiliary basis options like SARC/J or AutoAux can be found in Section 2.7.4.

Recontracted Ahlrichs Basis Sets

A list of available relativistically recontracted variants of the original Ahlrichs basis sets is given in Table 2.26. Refer to the section *Ahlrichs Basis Sets* for the origin of the legacy definitions with the prefix "old-".

Table 2.26: Relativistically recontracted Ahlrichs basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
DKH-SV(P)	H–Kr	DKH2	
DKH-SVP	H–Kr	DKH2	
DKH-TZV(P)	H–Kr	DKH2	
DKH-TZVP	H–Kr	DKH2	
DKH-TZVPP	H–Kr	DKH2	
DKH-QZVP	H–Kr	DKH2	
DKH-QZVPP	H–Kr	DKH2	
ZORA-SV(P)	H–Kr	ZORA	
ZORA-SVP	H–Kr	ZORA	
ZORA-TZV(P)	H–Kr	ZORA	
ZORA-TZVP	H–Kr	ZORA	
ZORA-TZVPP	H–Kr	ZORA	
ZORA-QZVP	H–Kr	ZORA	
ZORA-QZVPP	H–Kr	ZORA	
Legacy definitions (not recommended!)			
old-DKH-SV(P)	H–I	DKH2	
old-DKH-SVP	H–I	DKH2	
old-DKH-TZV(P)	H–I	DKH2	
old-DKH-TZVP	H–I	DKH2	
old-DKH-TZVPP	H–I	DKH2	
old-ZORA-SV(P)	H–I	ZORA	
old-ZORA-SVP	H–I	ZORA	
old-ZORA-TZV(P)	H–I	ZORA	
old-ZORA-TZVP	H–I	ZORA	
old-ZORA-TZVPP	H–I	ZORA	

Recontracted Karlsruhe def2 Basis Sets

A list of available adapted *DKH* and *ZORA* versions of the def2 basis sets (i.e., for the all-electron def2 basis sets) is given in Table 2.27. These basis sets retain the original def2 exponents but have only one contracted function per angular momentum (and hence are somewhat larger), with contraction coefficients suitable for the respective scalar relativistic Hamiltonian. These basis sets can be combined with the *SARC* and *SARC2* basis sets for the heavier elements.

Table 2.27: Relativistically recontracted Karlsruhe basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
DKH-def2-SVP	H–Kr	DKH2	
DKH-def2-SV(P)	H–Kr	DKH2	
DKH-def2-TZVP	H–Kr	DKH2	
DKH-def2-TZVP(-f)	H–Kr	DKH2	
DKH-def2-TZVPP	H–Kr	DKH2	
DKH-def2-QZVPP	H–Kr	DKH2	
ZORA-def2-SVP	H–Kr	ZORA	
ZORA-def2-SV(P)	H–Kr	ZORA	
ZORA-def2-TZVP	H–Kr	ZORA	
ZORA-def2-TZVP(-f)	H–Kr	ZORA	
ZORA-def2-TZVPP	H–Kr	ZORA	
ZORA-def2-QZVPP	H–Kr	ZORA	
Minimally augmented (scheme by Truhlar et al.[17])			
ma-DKH-def2-SVP	H–Kr	DKH2	
ma-DKH-def2-SV(P)	H–Kr	DKH2	
ma-DKH-def2-TZVP	H–Kr	DKH2	
ma-DKH-def2-TZVP(-	H–Kr	DKH2	
ma-DKH-def2-TZVPP	H–Kr	DKH2	
ma-DKH-def2-QZVPP	H–Kr	DKH2	
ma-ZORA-def2-SVP	H–Kr	ZORA	
ma-ZORA-def2-SV(P)	H–Kr	ZORA	
ma-ZORA-def2-TZVP	H–Kr	ZORA	
ma-ZORA-def2-TZVP(	H–Kr	ZORA	
ma-ZORA-def2-TZVPP	H–Kr	ZORA	
ma-ZORA-def2-QZVPP	H–Kr	ZORA	

SARC Basis Sets

Segmented all-electron relativistically contracted (SARC) basis sets for use with the DKH2 and ZORA Hamiltonians.[50, 51, 52, 53, 54, 55] A list of available SARC basis sets is given in Table 2.28.

Tip

Specifically for wavefunction-based calculations of lanthanide systems we recommend the more heavily polarized *SARC2 basis sets* [56].

Table 2.28: Relativistic SARC basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
`SARC-DKH-SVP`	Hf–Hg	DKH2	
`SARC-DKH-TZVP`	Rb–Rn, Ac–Lr	DKH2	
`SARC-DKH-TZVPP`	Rb–Rn, Ac–Lr	DKH2	
`SARC-ZORA-SVP`	Hf–Hg	ZORA	
`SARC-ZORA-TZVP`	Rb–Rn, Ac–Lr	ZORA	
`SARC-ZORA-TZVPP`	Rb–Rn, Ac–Lr	ZORA	

Note

SARC/J is the general-purpose Coulomb-fitting auxiliary for all SARC orbital basis sets.

SARC2 Basis Sets

SARC basis sets of valence quadruple-zeta quality for lanthanides, with NEVPT2-optimized (3g2h) polarization functions (SARC2).[56] Suitable for accurate calculations using correlated wavefunction methods. A list of available SARC2 basis sets is given in Table 2.29.

Table 2.29: Relativistic SARC2 basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	ECP	Comment
`SARC2-DKH-QZV`	La–Lu	DKH2	
`SARC2-DKH-QZVP`	La–Lu	DKH2	
`SARC2-ZORA-QZV`	La–Lu	ZORA	
`SARC2-ZORA-QZVP`	La–Lu	ZORA	

Note

Each basis set has a large dedicated /JK auxiliary basis set for simultaneous Coulomb and exchange fitting (cf. Table 2.35).

Karlsruhe x2c Basis Sets

For calculations with the X2C Hamiltonian, all-electron basis sets up to Rn are available.[57] The "-s" variants, e.g. `x2c-TZVPall-s`, are augmented with additional tight functions for NMR shielding calculations.[58] The "-2c" variants, e.g. `x2c-TZVPall-2c`, are intended for two-component calculations including spin-orbit coupling (**Note that two-component calculations are not implemented in *ORCA***). A list of available basis sets of this family is given in Table 2.30.

> **Tip**
>
> The `x2c/J` and `AutoAux` auxiliary basis set options can be used for these basis sets.

Table 2.30: Karlsruhe basis sets optimized for the x2c Hamiltonian.[57] RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
`x2c-SV(P)all`	H–Rn	X2C	
`x2c-SVPall`	H–Rn	X2C	
`x2c-TZVPall`	H–Rn	X2C	
`x2c-TZVPPall`	H–Rn	X2C	
`x2c-QZVPall`	H–Rn	X2C	
`x2c-QZVPPall`	H–Rn	X2C	
NMR shielding optimized[58]			
`x2c-SV(P)all-s`	H–Rn	X2C	
`x2c-SVPall-s`	H–Rn	X2C	
`x2c-TZVPall-s`	H–Rn	X2C	
`x2c-TZVPPall-s`	H–Rn	X2C	
`x2c-QZVPall-s`	H–Rn	X2C	
`x2c-QZVPPall-s`	H–Rn	X2C	
Two-component variants (no matching Hamiltonian yet!)			
`x2c-SV(P)all-2c`	H–Rn	SO-X2C	
`x2c-SVPall-2c`	H–Rn	SO-X2C	
`x2c-TZVPall-2c`	H–Rn	SO-X2C	
`x2c-TZVPPall-2c`	H–Rn	SO-X2C	
`x2c-QZVPall-2c`	H–Rn	SO-X2C	
`x2c-QZVPPall-2c`	H–Rn	SO-X2C	
`x2c-QZVPall-2c-s`	H–Rn	SO-X2C	
`x2c-QZVPPall-2c-s`	H–Rn	SO-X2C	

Relativistic Sapporo Basis Sets

A list of relativistic variants of the *Sapporo basis sets* that were optimized for the DKH3 Hamiltonian and finite nucleus are given in Table 2.31.

Table 2.31: Relativistic Sapporo basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
`Sapporo-DKH3-DZP-2`	K–Rn	DKH3	Optimized for DKH3 and finite nucleus
`Sapporo-DKH3-TZP-2`	K–Rn	DKH3	
`Sapporo-DKH3-QZP-2`	K–Rn	DKH3	

Relativistic Correlation-Consistent Basis Sets

A list of relativistic variants of the *correlation-consistent basis sets* is given in Table 2.32.

Table 2.32: Relativistic correlation-consistent basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
cc-pVDZ-DK	H–Ar, Sc–Kr	DKH2	
cc-pVTZ-DK	H–Ar, Sc–Kr, Y–Xe, Hf–Rn	DKH2	
cc-pVQZ-DK	H–Ar, Sc–Kr, In–Xe, Tl–Rn	DKH2	
cc-pV5Z-DK	H–Ar, Sc–Kr	DKH2	
cc-pVDZ-DK3	U	DKH3	For use with 3rd-order DKH
cc-pVTZ-DK3	U	DKH3	
cc-pVQZ-DK3	U	DKH3	
aug-cc-pVDZ-DK	H–Ar, Sc–Kr	DKH2	cc-pVDZ-DK augmented by diffuse functions
aug-cc-pVTZ-DK	H–Ar, Sc–Kr, Y–Xe, Hf–Rn	DKH2	cc-pVTZ-DK augmented by diffuse functions
aug-cc-pVQZ-DK	H–Ar, Sc–Kr, In–Xe, Tl–Rn	DKH2	cc-pVQZ-DK augmented by diffuse functions
aug-cc-pV5Z-DK	H–Ar, Sc–Kr	DKH2	cc-pV5Z-DK augmented by diffuse functions
cc-pwCVDZ-DK	H–Be, Na–Mg, Ca–Zn	DKH2	Equals cc-pVDZ-DK for H and He
cc-pwCVTZ-DK	H–Be, Na–Mg, Ca–Zn, Y–Xe, Hf–Rn	DKH2	Equals cc-pVTZ-DK for H and He
cc-pwCVQZ-DK	H–Be, Na–Mg, Ca–Zn, In–Xe, Tl–Rn	DKH2	Equals cc-pVQZ-DK for H and He
cc-pwCV5Z-DK	H–Be, Na–Mg, Ca–Zn	DKH2	Equals cc-pV5Z-DK for H and He
cc-pwCVDZ-DK3	U	DKH3	
cc-pwCVTZ-DK3	U	DKH3	
cc-pwCVQZ-DK3	U	DKH3	
aug-cc-pwCVDZ-DK	H–Be, Na–Mg, Sc–Zn	DKH3	cc-pwCVDZ-DK augmented by diffuse functions, equals aug-cc-pVDZ-DK for H and He
aug-cc-pwCVTZ-DK	H–Be, Na–Mg, Sc–Zn, Y–Xe, Hf–Rn	DKH2	cc-pwCVTZ-DK augmented by diffuse functions, equals aug-cc-pVTZ-DK for H and He
aug-cc-pwCVQZ-DK	H–Be, Na–Mg, Sc–Zn, In–Xe, Tl–Rn	DKH2	cc-pwCVQZ-DK augmented by diffuse functions, equals aug-cc-pVQZ-DK for H and He
aug-cc-pwCV5Z-DK	H–Be, Na–Mg, Sc–Zn	DKH2	cc-pwCV5Z-DK augmented by diffuse functions, equals aug-cc-pV5Z-DK for H and He

Relativistically Contracted ANO Basis Sets

The relativistic contracted ANO basis sets of Roos and coworkers were developed for the DKH2 Hamiltonian. The full list is given in Table 2.33.

Table 2.33: Relativistic contracted ANO basis sets. RH = relativistic Hamiltonian.

Basis Set	Elem.	RH	Comment
`ANO-RCC-Full`	H–Cm	DKH2	Complete ANO-RCC basis sets.
`ANO-RCC-DZP`	H–Cm	DKH2	Double-ζ contraction of `ANO-RCC-Full`.
`ANO-RCC-TZP`	H–Cm	DKH2	Triple-ζ contraction of `ANO-RCC-Full`.
`ANO-RCC-QZP`	H–Cm	DKH2	Quadruple-ζ contraction of `ANO-RCC-Full`.

2.7.4 Auxiliary Basis Sets

Resolution-of-the-idendity (RI) techniques can be used to speed up various types of calculations. Any RI method requires the definition of a reasonable auxiliary basis set in addition to the chosen *orbital basis set*. ORCA provides various built-in auxiliary basis set options for *`AuxJ`*, *`AuxJK`*, *`AuxC`*, and *`CABS`* and an automatic auxiliary basis set generation called *AutoAux*. Auxiliary basis sets can further be read from external files as described in Section 2.7.9.

> **Note**
>
> The distinction beweeen `AuxJ`, `AuxJK`, `AuxC`, and `CABS`, as well as how to correctly assign them, is *explained above*.

Coulomb-fitting auxiliary basis sets (`AuxJ`)

The auxiliary basis sets listed in Table 2.34 are suitable for the *RI-J*, *RIJDX/RIJONX*, and *RIJCOSX* approximations.

Table 2.34: Available Coulomb-fitting auxiliary basis sets.

Keyword	Elements	Comment
`def2/J`	H–Lr	Weigend's "universal" Coulomb-fitting basis suitable for all *def2* and *def* type basis sets. Assumes the use of ECPs beyond Kr (do not use with DKH/ZORA/X2C).
`def2-mTZVP/J`	H–Lr	
`def2-mTZVPP/J`	H–Lr	
`x2c/J`	H–Rn	Weigend's Coulomb-fitting basis for the all-electron *x2c-XVPall* basis sets
`SARC/J`	H–Rn, Ac–Lr	General-purpose Coulomb-fitting basis set for all-electron calculations. Consists of the decontracted def2/J up to Kr and of our own auxiliary basis sets for the rest of the periodic table. Appropriate for use in DKH or ZORA calculations with the *recontracted versions of the all-electron def2* basis sets (up to Kr) and the *SARC* basis sets for the heavier elements.

Coulomb- and exchange-fitting auxiliary basis sets (`AuxJK`)

The auxiliary basis sets listed in Table 2.35 are optimized for the *RI-JK* approximation. They can safely be used for *RI-J*, *RIJDX/RIJONX*, or *RIJCOSX*, but in that case they must be assigned to `AuxJ` via the `%basis` block.

Table 2.35: Available Coulomb- and exchange-fitting auxiliary basis sets.

Keyword	Elements	Comment
def2/JK	H–Rn	Coulomb+Exchange fitting for all *def2* basis sets
def2/JKsmall	H–Ra, Th–Lr	reduced version of def2/JK
cc-pVTZ/JK	H, B–F, Al–Cl, Ga–Br	For use with the respective cc-pVnZ orbital basis
cc-pVQZ/JK	H, B–F, Al–Cl, Ga–Br	
cc-pV5Z/JK	H, B–F, Al–Cl, Ga–Br	
aug-cc-pVTZ/JK	H, B–F, Al–Cl, Ga–Br	For use with the respective aug-cc-pVnZ orbital basis
aug-cc-pVQZ/JK	H, B–F, Al–Cl, Ga–Br	
aug-cc-pV5Z/JK	H, B–F, Al–Cl, Ga–Br	
SARC2-DKH-QZV/JK	La–Lu	
SARC2-DKH-QZVP/JK	La–Lu	
SARC2-ZORA-QZV/JK	La–Lu	
SARC2-ZORA-QZVP/JK	La–Lu	

Auxiliary basis sets for correlated methods (AuxC)

The available auxiliary basis sets suitable for post-(CAS)SCF dynamical electron correlation methods are listed in Table 2.36.

Table 2.36: Available auxiliary basis sets for correlated methods.

Keyword	Elements	Comment
def2-SVP/C	H–Rn	For use with the respective def2 orbital basis
def2-TZVP/C	H–Rn	
def2-TZVPP/C	H–Rn	
def2-QZVPP/C	H–Rn	
def2-SVPD/C	H–La, Hf–Rn	
def2-TZVPD/C	H–La, Hf–Rn	
def2-TZVPPD/C	H–La, Hf–Rn	
def2-QZVPPD/C	H–La, Hf–Rn	
cc-pVDZ/C	H–Ar, Ga–Kr	For use with the respective cc-pVnZ orbital basis
cc-pVTZ/C	H–Ar, Sc–Kr	
cc-pVQZ/C	H–Ar, Sc–Kr	
cc-pV5Z/C	H–Ar, Ga–Kr	
cc-pV6Z/C	H–He, B–Ne, Al–Ar	
aug-cc-pVDZ/C	H–He, Be–Ne, Mg–Ar, Ga–Kr	For use with the respective aug-cc-pVnZ orbital basis
aug-cc-pVTZ/C	H–He, Be–Ne, Mg–Ar, Sc–Kr	
aug-cc-pVQZ/C	H–He, Be–Ne, Mg–Ar, Sc–Kr	
aug-cc-pV5Z/C	H–Ne, Al–Ar, Ga–Kr	
aug-cc-pV6Z/C	H–He, B–Ne, Al–Ar	
cc-pwCVDZ/C	H–He, B–Ne, Al–Ar, Ga–Kr	For use with the respective cc-pwCVnZ orbital basis, equals
cc-pwCVTZ/C	H–He, B–Ne, Al–Ar, Sc–Kr	cc-pVnZ/C for H and He
cc-pwCVQZ/C	H–He, B–Ne, Al–Ar, Ga–Kr	
cc-pwCV5Z/C	H–Ne, Al–Ar	
aug-cc-pwCVDZ/C	H–He, B–Ne, Al–Ar, Ga–Kr	For use with the respective aug-cc-pwCVnZ orbital basis, equals
aug-cc-pwCVTZ/C	H–He, B–Ne, Al–Ar, Sc–Kr	aug-cc-pVnZ/C for H and He
aug-cc-pwCVQZ/C	H–He, B–Ne, Al–Ar, Ga–Kr	
aug-cc-pwCV5Z/C	H–Ne, Al–Ar	
cc-pVDZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	For use with the respective cc-pVnZ-PP orbital basis
cc-pVTZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
cc-pVQZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
aug-cc-pVDZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	For use with the respective aug-cc-pVnZ-PP orbital basis
aug-cc-pVTZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
aug-cc-pVQZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
cc-pwCVDZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	For use with the respective cc-pwCVnZ-PP orbital basis
cc-pwCVTZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
cc-pwCVQZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
aug-cc-pwCVDZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	For use with the respective aug-cc-pwCVnZ-PP orbital basis
aug-cc-pwCVTZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
aug-cc-pwCVQZ-PP/C	Cu–Kr, Y–Xe, Hf–Rn	
cc-pVDZ-F12-MP2Fit	H–Ar	For use with the respective cc-pVnZ-F12 orbital basis
cc-pVTZ-F12-MP2Fit	H–Ar	
cc-pVQZ-F12-MP2Fit	H–Ar	
cc-pVDZ-PP-F12-MP2Fit	Ga–Kr, In–Xe, Tl–Rn	For use with the respective cc-pVnZ-PP-F12 orbital basis
cc-pVTZ-PP-F12-MP2Fit	Ga–Kr, In–Xe, Tl–Rn	
cc-pVQZ-PP-F12-MP2Fit	Ga–Kr, In–Xe, Tl–Rn	
cc-pCVDZ-F12-MP2Fit	Li–Ar	For use with the respective cc-pCVnZ-F12 orbital basis
cc-pCVTZ-F12-MP2Fit	Li–Ar	

continues on next page

Table 2.36 – continued from previous page

Keyword	Elements	Comment
cc-pCVQZ-F12-MP2Fit	Li–Ar	

Complementary auxiliary basis sets for F12 (CABS)

The available CABS options for *F12 methods* are listed in Table 2.37.

Table 2.37: Available complementary auxiliary basis sets for F12 methods.

Keyword	Elements	Comment
cc-pVDZ-F12-CABS	H, B–Ne, Al–Ar	For use with the respective cc-pVnZ-F12 orbital basis
cc-pVTZ-F12-CABS	H, B–Ne, Al–Ar	
cc-pVQZ-F12-CABS	H, B–Ne, Al–Ar	
cc-pVDZ-F12-OptRI	H–Ar	Identical to the cc-pVnZ-F12-CABS basis above
cc-pVTZ-F12-OptRI	H–Ar	
cc-pVQZ-F12-OptRI	H–Ar	
cc-pVDZ-PP-F12-OptRI	Ga–Kr, In–Xe, Tl–Rn	For use with the respective cc-pVnZ-PP-F12 orbital basis
cc-pVTZ-PP-F12-OptRI	Ga–Kr, In–Xe, Tl–Rn	
cc-pVQZ-PP-F12-OptRI	Ga–Kr, In–Xe, Tl–Rn	
aug-cc-pVDZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	For use with the respective aug-cc-pVnZ-PP-F12 orbital basis
aug-cc-pVTZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	
aug-cc-pVQZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	
aug-cc-pV5Z-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	
cc-pCVDZ-F12-OptRI	Li–Ar	For use with the respective cc-pCVnZ-PP-F12 orbital basis
cc-pCVTZ-F12-OptRI	Li–Ar	
cc-pCVQZ-F12-OptRI	Li–Ar	
aug-cc-pwCVDZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	For use with the respective aug-cc-pwCVnZ-PP-F12 orbital basis
aug-cc-pwCVTZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	
aug-cc-pwCVQZ-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	
aug-cc-pwCV5Z-PP-OptRI	Cu–Zn, Ag–Cd, Au–Hg	

Automatic Generation of Auxiliary Basis Sets (AutoAux)

If no auxiliary basis set is available for your chosen orbital basis set, one can be generated automatically by ORCA using the keyword `AutoAux`. This is specified as any other fitting basis set: as a value to the `AuxJ/AuxJK/AuxC` variables in the `%basis` block or as a separate keyword in the simple input line (in which case all three Aux slots are populated with identical fitting basis sets). `AutoAux` can also be assigned to individual elements or atoms – see sections *Assigning or Adding Basis Functions to an Element* and *Assigning or Adding Basis Functions to Individual Atoms*. The generated basis sets can be used for Coulomb, exchange and correlation fitting and are as accurate as the optimized auxiliary basis sets at the cost of being up to twice as large.[59] The available settings for `AutoAux` are given in Table 2.41 with their default values.

> **Note**
>
> The generation procedure in ORCA 3.1 was significantly different and does not produce the same results! For compatibility, the old version is still accessible via the setting `OldAutoAux true` in the `%basis` block.

Note that if the orbital basis set contains diffuse functions, as is the case for the aug-cc-pVXZ sets, the AutoAux fitting basis may contain (near-)linear dependencies. In this case, the Cholesky decomposition of the Coulomb metric will fail and the program will likely crash. One may print the offending auxiliary basis using `!PrintBasis` and manually remove the most diffuse s- and/or p-functions, which will usually resolve the problem. An alternative, automatic solution is also implemented – see section *Removal of Redundant Basis Functions*.

2.7.5 Effective Core Potentials

Starting from version 2.8.0, ORCA features effective core potentials (ECPs). They are a good alternative to scalar relativistic all-electron calculations if heavy elements are involved. This pertains to geometry optimizations and energy calculations but may not be true for property calculations.

In order to reduce the computational effort, the usually highly contracted and chemically inert core basis functions can be eliminated by employing ECPs. ECP calculations comprise a "valence-only" basis and thus are subject to the frozen core approximation. Contributions due to the core orbitals are accounted for by an effective one-electron operator U^{core} which replaces the interactions between core and valence electrons and accounts for the indistinguishability of the electrons. Its radial parts $U_l(r)$ are generally expressed as a linear combination of Gaussian functions, while the angular dependence is included through angular momentum projectors $|S_m^l\rangle$.

$$U^{\text{core}} = U_L(r) + \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \left|S_m^l\right\rangle \left[U_l(r) - U_L(r)\right] \left\langle S_m^l\right|$$

$$U_l = \sum_k d_{kl} r^{n_{kl}} \exp(-\alpha_{kl} r^2)$$

The maximum angular momentum L is generally defined as $l_{\text{max}}^{\text{atom}} + 1$. The parameters n_{kl}, α_{kl} and d_{kl} that are necessary to evaluate the ECP integrals have been published by various authors, among them the well-known Los Alamos (LANL) [41] and Stuttgart–Dresden (SD) [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105] parameter sets. Depending on the specific parametrization of the ECP, relativistic effects can be included in a semiempirical fashion in an otherwise nonrelativistic calculation. Introducing U^{core} into the electronic Hamiltonian yields two types of ECP integrals, the local (or type-1) integrals that arise because of the maximum angular momentum potential U_L and the semi-local (or type-2) integrals that result from the projected potential terms. The evaluation of these integrals in ORCA proceeds according to the scheme published by Flores-Moreno et al.[106].

A selection of ECP parameters and associated basis sets is directly accessible in ORCA through the internal ECP library (see Table 2.38 for a listing of keywords).

Table 2.38: Overview of library keywords for ECPs and associated basis sets available in ORCA.

ECP keyword	Core size[Page 83, 1]	Elements	Valence basis sets
Recommended			
`def-ECP`	78	Fr–Ra	Karlsruhe def2 basis sets: def-SV(P), def-SVP, def-TZVP, def-TZVPP, ma-def-TZVP
	60	Ac–Lr	MINIX
`def2-ECP`	28	Rb–Xe	Karlsruhe basis sets: def2-SVP, def2-TZVP, etc. def2-SVPD, def2-TZVPD, etc. ma-def2-SVP, ma-def2-TZVP, etc.
	46	Cs–La	
	28	Ce–Lu	
	60	Hf–Rn	
`SK-MCDHF-RSC`	10	Ca, Cu–Kr	Correlation-consistent basis sets: cc-pVnZ-PP, aug-cc-pVnZ-PP, cc-pCVnZ-PP, aug-cc-pCVnZ-PP, cc-pwCVnZ-PP, aug-cc-pwCVnZ-PP (n = D, T, Q, 5) cc-pVnZ-PP (n = D, T, Q)
	28	Sr–Xe	
	46	Ba	
	60	Hf–Rn	
	78	Ra	
	60	U	
`HayWadt`[2]	10	Na–Cu	LANL basis sets: LANL2DZ, LANL2TZ, LANL2TZ(f), LANL08, LANL08(f)
	18	Zn	
	28	Ga–Ag	
	36	Cd	
	46	In–La	
	60	Hf–Au	
	68	Hg–Tl	
	78	Pb–Bi, U–Pu	
`dhf-ECP`	28	Rb–Xe	Karlsruhe dhf basis sets: dhf-SVP, dhf-TZVP, etc.
	46	Cs–Ba	
	60	Hf–Rn, U	
`vDZP-ECP`	2	B–Mg	vDZP *uniquely compiled for the use with vDZP*
	10	Al–Zn	
	28	Ga–Cd	

continues on next page

Table 2.38 – continued from previous page

ECP keyword	Core size[1]	Elements	Valence basis sets
	46	In–Lu	
	60	Hf–Hg	
	78	Tl–Rn	
CRENBL-ECP	2	Li–Mg	CRENBL
	10	Al–Zn	
	28	Rb–Cd	
	36	In–Xe	
	46	Cs–La	
	54	Ce–Lu	
	60	Hf-Hg	
	68	Tl-Rn	
	78	Fr-Ts	
	92	Og	
Legacy definitions			
def2-SD	28,MWB	Rb–Cd	
	28,MDF[3]	In–Xe	
	46,MWB	Cs–La	
	60,MWB	Hf–Pt	
	60,MDF[4]	Au–Rn	
def-SD	28,MWB	Rb–Cd	
	46,MWB	In–La	
	28,MWB	Ce–Lu	
	60,MWB	Hf–Pt	
	60,MDF[4]	Au, Hg, Rn	
	78,MWB	Tl–At	
	78,MDF	Fr, Ra	
	60,MWB	Ac–Lr	
SDD	2,SDF	Li, Be	
	2,MWB	B–Ne	
	10,SDF	Na, Mg	
	10,MWB	Al–Ca	
	10,MDF	Sc–Ni	
	10,MWB	Cu-Zn	
	28,MWB	Ga–Sr	
	28,MHF	Y–Cd	
	28,MDF	Ge–Br, Rb–Xe	
	46,MWB	In–Ba	
	28,MWB	La–Lu	
	60,MWB	Hf–Hg	
	78,MWB	Tl–Rn	
	60,MWB	Ac–Lr	
LANL1	10	Na–Ar	
	18	K–Zn	
	28	Ga–Kr	
	36	Rb–Cd	
	46	In–Xe	
	54	Cs–La	
	68	Hf–Tl	
	78	Pb, Bi	
LANL2	10	K–Cu	
	28	Rb–Ag	
	46	Cs–La	
	60	Hf–Au	

Note

Requesting some basis sets automatically assigns the matching ECP (except when using the `NewGTO` keyword): for example, "def2" basis sets use the `def2-ECP`. For others, see the respective *basis set table entries*.

The simplest way to assign ECPs is by using the ECP keyword within the simple input line. The ECP keyword itself

[1] Where applicable, reference method and data are given (S: single-valence-electron ion; M: neutral atom; HF: Hartree–Fock; WB: quasi-relativistic; DF: relativistic).

[2] Corresponds to LANL2 and to LANL1 where LANL2 is unavailable.

[3] I: OLD-SD(28,MDF) for compatibility with TURBOMOLE.

[4] Au, Hg: OLD-SD(60,MDF) for compatibility with TURBOMOLE.

assigns *only* the effective core potential, not a valence basis set! As an example for an explicitly named ECP you could use

```
! def2-TZVP def2-SD
```

This would assign the def2-SD ECP according to the definition given in the table above. Without the def2-SD keyword ORCA would default to def2-ECP.

Assignment of ECPs can be done within the `%basis` block using the `ECP` and `NewECP` keywords, as in the following example:

```
%basis
  ECP       "def2-ECP"     # All elements (for which the ECP is defined)
  NewECP Pt "def2-SD" end  # Different ECP for Pt
end
```

A variant of the `NewECP` keyword can be used for individual atoms inside the geometry definition:

```
* xyz ...
  ...
  S   0.0  0.0  0.0  NewECP "SDD" end
  ...
*
```

Note that these keywords also only affect the ECP and not the valence basis set!

In case the basis set for an element/atom has been changed using the `NewGTO` keyword (see sections *Assigning or Adding Basis Functions to an Element* and *Assigning or Adding Basis Functions to Individual Atoms* above) it may be necessary to remove the ECP from that element/atom. This can be done with the `DelECP` keyword in the `%basis` block or coordinates input, respectively:

```
! LANL2DZ                   # Uses HayWadt ECPs by default, starting from Na
%basis
  NewGTO S "def2-TZVP" end  # All-electron up to Kr
  DelECP S                  # Remove HayWadt ECP
end
* xyz ...
...
Cu   0.0  0.0  0.0
  DelECP                    # Remove HayWadt ECP
  NewGTO "def2-QZVPP" end   # All-electron up to Kr
...
*
```

To remove all ECPs loaded by default (e.g. in case no global basis set is chosen) you can use the `!NoECP` simple keyword.

Manual Input of ECP Parameters

To manually specify ECP parameters, the `NewECP` keyword is followed by the element for which an ECP is to be entered, the number of core electrons to be replaced (N_core) and the maximum angular momentum (lmax). The ECP specification is finished by giving the definitions of the individual shells that constitute the angular dependent potentials U_l.

```
%basis
    NewECP <element>
      N_core <number of core electrons>
      lmax <max. angular momentum>
      [shells]
    end
end
```

For each ECP shell, first the angular momentum l has to be given, followed by the number of primitives. The primitives themselves are then specified by giving a running index and the respective tuple of exponent a_{kl}, expansion coefficient d_{kl} and radial power n_{kl}.

```
# ECP shell
      l <number of primitives>
      1    a1l    d1l    n1l
      2    a2l    d2l    n3l
      ...
```

As an example, consider the SD(10,MDF) for Vanadium. The name indicates a Stuttgart–Dresden type ECP that replaces 10 core electrons and is derived from a relativistic calculation for the neutral atom. It consists of 4 shells with angular momentum s, p, d, and f. Note that the f shell has an expansion coefficient of 0.0 and thus will not contribute at all to this effective core potential. This is typical for all SD potentials (but may be different for program packages like TURBOMOLE that do not support arbitrary angular momentum with respect to the ECP and therefore use recontractions of the original parameter sets).

```
%basis
  # ECP SD(10,MDF) for V
  # M. Dolg, U. Wedig, H. Stoll, H. Preuss,
  # J. Chem. Phys. 86, 866 (1987).
  NewECP V
    N_core 10
    lmax f
    s 2
      1      14.4900000000    178.4479710000  2
      2       6.5240000000     19.8313750000  2
    p 2
      1      14.3000000000    109.5297630000  2
      2       6.0210000000     12.5703100000  2
    d 2
      1      17.4800000000    -19.2196570000  2
      2       5.7090000000     -0.6427750000  2
    f 1
      1       1.0000000000      0.0000000000  2
  end
end
```

ECPs and Ghost Atoms

When ghost atoms are defined in the input (see section *Special definitions*), ECPs are **not** added to these atoms by default. If that is somehow needed, please add `GhostECP true` under the `%basis` block.

```
%basis
 GhostECP true
 AllowGhostECP true # synonym
end
```

ECP Embedding

Computations on cluster models sometimes require the presence of embedding potentials in order to account for otherwise neglected repulsive terms at the border [107]. In order to simplify these kind of calculations with ORCA the ECP embedding can be accomplished quite easily:

```
*xyz ...
# atom>  charge x    y    z    optional ECP declaration
Zr>      4.0    0.0  0.0  0.0  NewECP "SDD" end
...
*
```

The declaration of such a coreless ECP center takes place in the coordinates section by appending a bracket ">" to the element symbol. Note that embedding ECPs are treated as point charges in ORCA, so the charge has to be given next. The coordinates of the coreless ECP center have to be specified as usual and may be followed by an optional ECP assignment. In general, calculations that employ an ECP embedding procedure should be single point calculations. However if the need arises to perform a geometry optimization, make sure to set up explicit Cartesian constraints for the coreless ECP centers.

2.7.6 Assigning or Adding Basis Functions to an Element

In order to assign a new basis set to a given element, use:

```
%basis
  NewGTO 8    # New basis for oxygen.
# NewGTO O    # This works as well.
   S   3      # s-shell
    1  910.10034975         0.03280967
    2  137.19711335         0.23422391
    3   30.85279077         0.81490980
   0   2      # also an s-shell
    1     1.72885887         0.27389659
    2     0.39954770         0.79112437
   P   1
    1     8.35065975         1.00000000
  end
end
```

For simplicity and consistency the input format is the same as that used in the *basis set files*. In this format, the first line carries the angular momentum of the shell to be added – either as an integer, or as a label (s, p, d, f, g, h, i, k) – and the number of primitives. Then for each primitive one line follows which has (a) the index of the primitive (1, 2, 3, …) (b) the exponent of the primitive and (c) the contraction coefficient (unnormalized). There also is the possibility to include a `SCALE X` statement after the number of primitives in the first line to indicate that the basis function exponents should be scaled.

Warning

- ORCA always uses spherical harmonic Gaussian functions.
- Angular momentum 7 is labeled as "k" – there are no j-shells in accepted spectroscopic conventions.
- Combined s- and p-shells are sometimes labelled as "L-shells" in other programs. This is not supported in the `NewGTO` format and to avoid confusion, shells with angular momentum 8 can only be specified with a number, and not with the label "l".

In order to add basis functions to the basis of a given element (for example because you do not like the standard polarization functions) use `AddGTO` instead of `NewGTO`. In `NewGTO` or `AddGTO` you can also use the nicknames of internally stored basis sets. An example is:

```
%basis
  NewGTO 8    # new basis for oxygen
   "6-31G"
    D  1
    1  0.4  1.0
  end
end
```

In this example the 6-31G basis is assigned to oxygen and in addition a polarization function with exponent 0.4 is added to the oxygen basis.

Note that the `NewGTO` keyword does not change the ECP for the given element – you must use `NewECP` or `DelECP` (see section *Effective Core Potentials*).

A similar mechanism was established for the auxiliary basis sets in RI calculations:

```
%basis
  NewAuxJGTO 8    # new auxiliary basis for oxygen
    s  1
    1  350  1.0
    ... etc
  end
  AddAuxJGTO 8   # add a shell to the auxiliary basis for
               # oxygen
    D  1
    1  0.8  1.0
  end
end
```

New basis functions can be specifically assigned to any auxiliary basis sets. The keywords `NewAuxCGTO`, `AddAuxCGTO`, `NewAuxJKGTO`, `AddAuxJKGTO`, `NewCABSGTO`, `AddCABSGTO` are used in the same way. The keywords `NewAuxGTO` and `AddAuxGTO` are the same as `NewAuxJGTO` and `AddAuxJGTO`, that is, they only influence the Coulomb auxiliary basis (`AuxJ`)!

2.7.7 Assigning or Adding Basis Functions to Individual Atoms

Sometimes you may want to not treat all atoms of the same element with the same basis set but to assign a specific basis set to a specific atom in the molecules. This is also possible in ORCA and takes place in the coordinate section (`%coords`, `*xyz`, etc.). The format is the same as described above. An example may help to make things clear:

```
*int 0 1
  C  0 0 0 0.00   0.0   0.00
   AddGTO
     D  1
      1  1.0  1.0
   end
  O  1 0 0 1.13   0.0   0.00
   NewGTO
     "6-311G"
     D  1
      1  1.2  1.0
   end
*
```

In this example an extra d-shell with exponent 1.0 is added to the first carbon atom and the basis for the oxygen atom is changed to `6-311G` with an extra d-function of exponent 1.2 added.

Analogously, `AUX` basis functions can be assigned or added to individual atoms using the keywords `NewAuxJGTO`, `AddAuxJGTO`, `NewAuxCGTO`, `AddAuxCGTO`, `NewAuxJKGTO`, `AddAuxJKGTO`, `NewCABSGTO`, `AddCABSGTO`.

A note on the use of `AutoAux`: if you change the basis set on a given atom and want to generate a fitting basis, you have to specify it again in the coordinates section, even if `AutoAux` is already present in the simple input line or in the `%basis` block. For example:

```
! def2-SVP def2/JK
%basis
  NewAuxJKGTO H
    "AutoAux"
  end
end
*xyz 0 1
  O  0.00 0.00 0.00
  H -0.25 0.93 0.00
  H  0.96 0.00 0.00
```

(continues on next page)

(continued from previous page)

```
   AddGTO
     P 1
      1 1.6 1.0
     D 1
      1 1.0 1.0
   end
   NewAuxJKGTO
     "AutoAux"
   end
*
```

Here the oxygen atom is assigned the def2-SVP basis and the def2/JK fitting basis, the first hydrogen atom is assigned the def2-SVP basis and an automatically generated fitting basis and the second hydrogen atom is assigned the def2-SVP basis with two additional polarization functions and a larger automatically generated fitting basis that accounts for these functions.

Tip

When assigning custom basis sets it is always a good idea to print the basis set information (`%output print[p_basis] 2 end` or simply `!PrintBasis`) and check that everything is correct.

2.7.8 Assigning Basis Sets and ECPs to Fragments

In multi-level or QM/QM calculations it may be convenient to assign different basis sets to different *fragments*. This can be done with the keywords `FragBasis`, `FragAuxJ`, `FragAuxJK`, `FragAuxC`, `FragCABS`, and `FragECP` in the `%basis` block, followed by the number of the fragment (numbering starts at 1!) and a standard basis set or ECP from the ORCA library (see Section 2.7.2 and Table 2.38). Note that unlike the `NewGTO` keyword, `FragBasis` also changes the ECP, if applicable. Fragment basis sets will overload the global or element-specific (Section 2.7.6) choice but can be overloaded for individual atoms (Section 2.7.7). If AutoAux is requested for a fragment, it will be generated for the actual orbital basis set chosen for each atom, even if it is changed in the coordinates section. However, if AutoAux was requested for an element or in the simple input, the auxiliary basis will be generated before the fragment basis is assigned (which is not desired), therefore AutoAux must be requested again for the fragment. An example is given below:

```
! PrintBasis BP86 NoIter
! def2-SVP def2/J
%basis
  FragBasis 1 "def2-TZVP"
  FragBasis 2 "cc-pVTZ-PP"
  FragAuxJ  2 "AutoAux"
  FragECP   3 "SK-MCDHF-RSC"
  FragAuxJ  3 "def2/JK"
end
*xyz 0 1
  H(1) 0 0 0
  I(1) 0 0 1.6
  H(2) 0 5 0   NewGTO "cc-pVTZ" end
  I(2) 0 5 1.6
  H(3) 5 0 0
  I(3) 5 0 1.6
*
# Final basis sets:
# Atom Basis      ECP          AuxJ
# 0H   def2-TZVP  def2-ECP     def2/J
# 1I   def2-TZVP  def2-ECP     def2/J
# 2H   cc-pVTZ    -            AutoAux(cc-pVTZ)
# 3I   cc-pVTZ-PP SK-MCDHF-RSC AutoAux(cc-pVTZ-PP)
```

(continues on next page)

(continued from previous page)

```
# 4H   def2-SVP   -            def2/JK
# 5I   def2-SVP   SK-MCDHF-RSC def2/JK
```

It is also possible to read fragment-specific basis sets from a file. The syntax is analogous, using the keywords `ReadFragBasis`, `ReadFragAuxJ`, `ReadFragAuxJK`, `ReadFragAuxC`, `ReadFragCABS`, and `ReadFragECP`. In this case, the input string is expected to be an existing basis set file in GAMESS-US format (see section *Reading Basis Sets from a File*). All other details above (e.g., regarding ECPs and AutoAux) also apply here.

Note

- Details regarding the assignment of fragments can be found in *Fragment Specification* section.

2.7.9 Reading Basis Sets from a File

By using the variables `GTOName`, `GTOAuxJName`, `GTOAuxJKName`, `GTOAuxCName`, and `GTOCABSName` (`GTOAuxName` is a synonym for `GTOAuxJName`) a basis set can be read from an ASCII file. In this way you can construct or modify your favorite standard basis set and load it easily into the program.

```
%basis
  GTOName      = "MyBasis.bas"      # read orbital basis
  GTOAuxJName  = "MyAuxJBasis.bas"  # read Coulomb-fitting basis
  GTOAuxJKName = "MyAuxJKBasis.bas" # read Coulomb- and exchange-fitting basis
  GTOAuxCName  = "MyAuxCBasis.bas"  # read correlation-fitting basis
  GTOCABSName  = "MyCABSBasis.bas"  # read complementary auxiliary basis set
end
```

A word of caution: under Windows, backslashes directory assignments must be given twice to be correctly understood! The format is that used for "GAMESS-US" in the EMSL library [108]. To give an example of what this format looks like here is a part of the `3-21GSP` basis of Buenker and coworkers [109, 110]:

```
!                          lines in the beginning with '!' or '#' are comments
! BASIS="3-21GSP"
!Elements                             References
!--------                             ----------
! H - Ne: A.V. Mitin, G. Hirsch, R. J. Buenker, Chem. Phys. Lett. 259, 151 (1996)
! Na - Ar: A.V. Mitin, G. Hirsch, R. J. Buenker, J. Comp. Chem. 18, 1200 (1997).
!
$DATA         ! Optional
HYDROGEN      ! (3s) -> [2s]      Element symbols are also recognized
 S   2
  1           4.50036231         0.15631167
  2           0.68128924         0.90466909
 S   1
  1           0.15137639         1.00000000
CARBON        ! (6s,3p) -> [3s,2p]
 S   3
  1         499.24042249         0.03330322
  2          75.25419194         0.23617745
  3          16.86538669         0.81336259
 L   2        ! L shells are a s and a p shell with identical exponents
  1           0.89739483         0.24008573         0.46214684
  2           0.21746772         0.81603757         0.66529098
 L   1
  1           4.52660451         1.00000000         1.00000000
$END          ! Optional
```

The file format for the auxiliary basis sets is exactly the same. Basis sets can be also exported in GAMESS-US

format by the `orca_exportbasis` utility (section *orca_exportbasis*). Note that in order to read basis sets printed by ORCA (using `!PrintBasis`), the `NewGTO` and `end` keywords must be removed.

Warning

- Angular momentum 7 is labeled as "k" – there are no j-shells in accepted spectroscopic conventions.
- To avoid confusion with combined s- and p-shells, shells with angular momentum 8 can only be specified with a number, and not with the label "l".

2.7.10 Linear Dependence

The previous sections describe the assessment of a desired molecular basis set from appropriately parametrized functions at various locations within the molecule (normally centered on atoms). The parametrization of these functions is such that the chance for redundancy is minimal. Since however, one is limited to work with finite numerical precision, and furthermore these parameters also depend on the molecular geometry, redundancies cannot be completely eliminated in advance. Redundancy means that the subspace spanned by the given basis functions at given values of parameters (including geometry), can be identically spanned by a smaller number of *linear independent* basis functions. Linear dependent (redundant) function sets however may cause numerical instabilities. Linear dependence is normally identified by searching for zero eigenvalues of the overlap matrix. Note that the inverse of the overlap (or related matrices) are used for orthogonalization purposes, and it follows that if near zero eigenvalues are not treated properly, the inverse becomes ill-defined, and the SCF procedure numerically unstable.

From the previous discussion, it is evident that the crucial parameter for curing linear dependence is the threshold below which an overlap eigenvalue is considered zero. This parameter may be changed using the following keyword

```
%scf
  sthresh 1e-6 # default 1e-7
end
```

Although there is no strict limit for the value of the above parameter, it should reasonably be somewhere between 1e-5 and 1e-8 (the default is 1e-7). One may get away with 1e-9 or perhaps even lower without convergence problem, but there is a risk that the result is contaminated with noise caused by the near zero vectors. In difficult cases, an 1e-6 threshold was often found to work smoothly, and above that one risks throwing away more and more functions, which also influence comparability of results with other calculations. To monitor the behavior of the small eigenvalues, one should look for the following block in the output

```
Diagonalization of the overlap matrix:
Smallest eigenvalue                        ... -1.340e-17
Time for diagonalization                   ...    0.313 sec
Threshold for overlap eigenvalues          ... 1.000e-07
Number of eigenvalues below threshold      ... 1
Smallest eigenvalue above threshold        ... 6.013e-07
Time for construction of square roots      ...    0.073 sec
Total time needed                          ...    0.387 sec
```

Here, the smallest eigenvalue is printed, along with the currently used overlap threshold, and the number of functions below this (which will be dropped). It is a recommended consistency check to look for an equal number of zero entries among orbital energies once the SCF procedure converged. Note that for functions belonging to zero eigenvalues *no level shifts* are applied!

In case that redundant vectors were removed from the basis, `! MORead NoIter` should only be used in conjunction with the same `SThresh` as in the original calculation, otherwise the results will be inconsistent. `! MORead` may still be used together with a change in `SThresh`, but a few SCF iterations will be required.

Automatic Adjustments for Near Linear-Dependent Cases

Starting from ORCA6, there is now a keyword called DiffSThresh, which controls an automatic tightening of the integral cutoff parameters Thresh and TCut in case small eigenvalues of the overlap matrix are found. We found this to be important in some calculations using diffuse basis, and these parameters are set to a minimum value of `Thresh=1e-12` and `TCut=1e-13` in case the "Smallest eigenvalue" shown above gets below that number. If the cutoffs are already tighter than that, for instance when using `!VeryTightSCF`, than nothing will happen.

We found empirically that these are safe numbers to mitigate noise and increase the robustness of the SCF procedure, thus they are enforced by default. The default is 1e-6 and this can be turned off by setting `%SCF DiffSThresh -1 END` on the input in case you don't want this automatic adjustment to happen.

Removal of Redundant Basis Functions

While the approach described above is usually successful in removing linear dependencies from the orbital basis set, the auxiliary basis used in RI is not orthogonalized the same way. Instead, the RI linear equation system is solved using a Cholesky decomposition (CD) of the auxiliary basis Coulomb metric. If the auxiliary basis is redundant, the CD fails and the program usually aborts. One simple solution implemented in ORCA is to perform a pivoted Cholesky decomposition (PCD) of the metric, terminating at a given threshold. Then, the shells contributing to the nullspace are removed from the basis at the beginning of the calculation. This can be requested for any of the basis sets using either the overlap or the Coulomb metric. It is most appropriate for the AuxJ/AuxJK/AuxC basis using the Coulomb metric. The truncated basis can be examined using the `!PrintBasis` keyword. Often, functions may be removed for some atoms of a given element, but kept for others. As long as the threshold is low enough, i.e. only truly redundant functions are removed, this should not affect the molecular symmetry of the results.

```
%basis
  PCDTrimBas   Overlap # Trim the orbital basis in the overlap metric
  PCDTrimAuxJ  Coulomb # Trim the AuxJ basis in the Coulomb metric
  PCDTrimAuxJK Coulomb # Trim the AuxJK basis in the Coulomb metric
  PCDTrimAuxC  Coulomb # Trim the AuxC basis in the Coulomb metric
  PCDThresh    -1      # Threshold for the PCD: chosen automatically if <0
end
```

2.7.11 Which Methods Need Which Basis Sets?

ORCA offers a variety of methods and a large choice of *orbital* and *auxiliary* basis sets to go with them. Pure (GGA or meta-GGA) *DFT* functionals only require the calculation of Coulomb integrals, while *hybrid DFT*, HF (and by extension, all post-HF electron correlation methods, such as *MP2* and *coupled cluster*), as well as *CASSCF* (and *NEVPT2*), require the calculation of Coulomb and exchange integrals.

- An orbital basis set (`<basis>`) is always needed for these methods.
- If RI is used for Coulomb integrals (*RI-J*, *RIJDX/RIJONX*, *RIJCOSX*), `AuxJ` is needed (usually `<basis>/J` or `def2/J`).
- If RI is also used for exchange integrals (*RI-JK*), `AuxJK` is needed instead (usually `<basis>/JK` or `def2/JK`).
- If RI is used for integral generation in post-SCF correlation methods, as in *RI-MP2* (including *double-hybrid DFT*), *DLPNO-MP2*, and *DLPNO-CC*, `AuxC` is also needed (usually `<basis/C>`).
- In *F12 methods*, a specialized orbital basis is used (`<basis>-F12`) and `CABS` is needed in addition (usually `<basis>-F12-CABS` or `<basis>-F12-OptRI`).

An overview of auxiliary basis requirements for an inexhaustive list of methods and approximations is given in Table 2.39.

Table 2.39: Simple input keywords for basis sets and ECPs.

Method	Approximation	Basis sets
HF	NoRI (default)	<basis>
HF	RIJONX or RIJCOSX	<basis> + <basis>/J
HF	RI-JK	<basis> + <basis>/JK
pure DFT	RI (default)	<basis> + <basis>/J
hybrid DFT	NoRI	<basis>
hybrid DFT	RIJCOSX (default)	<basis> + <basis>/J
hybrid DFT	RI-JK	<basis> + <basis>/JK
CASSCF/NEVPT2		<basis>
CASSCF/NEVPT2	RI-JK	<basis> + <basis>/JK
CASSCF/NEVPT2	RIJCOSX	<basis> + <basis>/J + <basis>/C
CASSCF/NEVPT2	TrafoStep RI	<basis> + <basis>/JK or <basis>/C
NEVPT2-F12	TrafoStep RI	<basis>-F12 + <basis>-F12-CABS + <basis>/JK or <basis>/C
TDDFT		<basis>
MP2		<basis>
RI-MP2		<basis> + <basis>/C
RI-MP2	RI-JK	<basis> + <basis>/C + <basis>/JK
F12-MP2		<basis>-F12 + <basis>-F12-CABS
F12-RI-MP2		<basis>-F12 + <basis>-F12-CABS + <basis>/C
DLPNO-MP2		<basis> + <basis>/C
DLPNO-MP2	RIJCOSX	<basis> + <basis>/C + <basis>/J
F12-DLPNO-MP2		<basis>-F12 + <basis>-F12-CABS + <basis>/C
CCSD		<basis>
RI-CCSD		<basis> + <basis>/C
DLPNO-CCSD		<basis> + <basis>/C
DLPNO-CCSD	RIJCOSX	<basis> + <basis>/C + <basis>/J
F12-CCSD		<basis>-F12 + <basis>-F12-CABS
F12-RI-CCSD		<basis>-F12 + <basis>-F12-CABS + <basis>/C
F12-RI-CCSD	RI-JK	<basis>-F12 + <basis>-F12-CABS + <basis>/C + <basis>/JK

2.7.12 Keywords

Table 2.40: Simple input keywords for basis sets and ECPs.

Keyword	Description
<BasisName>	Assign the respective *orbial basis set* to all elements
<AUXJName>	Assign the respective *AuxJ basis set* to all elements
<AUXJKName>	Assign the respective *AuxJK basis set* to all elements
<AUXCName>	Assign the respective *AuxJC basis set* to all elements
<CABSName>	Assign the respective *CABS* to all elements
AutoAux	Automatically generate AuxJ, AuxJK, and AuxC auxiliary basis sets (see Section 2.7.4)
<ECPName>	Assign the respective ECP to all elements for which it is defined
NoECP	Remove the default ECP
Decontract	Decontract all (orbital and auxiliary) basis sets
DecontractBas	Decontract the orbital basis sets
NoDecontractBas	Do not decontract the basis set
DecontractAuxJ	Decontract the AuxJ basis set
NoDecontractAuxJ	Do not decontract the AuxJ basis
DecontractAuxJK	Decontract the AuxJK basis set
NoDecontractAuxJK	Do not decontract the AuxJK basis
DecontractAuxC	Decontract the AuxC basis set
NoDecontractAuxC	Do not decontract the AuxC basis

Table 2.41: %basis block input keywords for basis sets and ECPs.

Keyword	Options	Description
Basis	"<BasisName>"	Define the *orbital basis set*
AuxJ	"<AuxName>"	Define the *J auxiliary basis set*
AuxJK	"<AuxName>"	Define the *JK auxiliary basis set*
AuxC	"<AuxName>"	Define the *correlation auxiliary basis set*
CABS	"<CABSName>"	Define the *complementary auxiliary basis set* for F12 calculations
ECP	"<ECPName>"	Assign the respective *ECP* to all elements for which it is available
GhostECP	false	Activate ECPs on ghost atoms
AllowGhostECP	false	Equivalent to GhostECP
Decontraction options		
Decontract	false	If true, decontract all basis sets
DecontractBas	false	If true, decontract the orbital basis set
DecontractAuxJ	false	If true, decontract the AuxJ basis set
DecontractAuxJK	false	If true, decontract the AuxJK basis set
DecontractAuxC	false	If true, decontract the AuxC basis set
DecontractCABS	true	If false, do not decontract the CABS
Setting basis sets for elements (see Section 2.7.6)		
NewGTO	<Element> "<BasisName>" <shells> End	Define new Basis for element via built-in name and/or custom shells
AddGTO	<Element> <shells> End	Add GTO shells to basis for element
NewAuxJGTO	<Element> "<AuxName>" <shells> End	Define new AuxJ set for element via built-in name and/or custom shells
AddAuxJGTO	<Element> <shells> End	Add GTO shells to AuxJ for element
NewAuxJKGTO	<Element> "<AuxName>" <shells> End	Define new AuxJK set for element via built-in name and/or custom shells
AddAuxJKGTO	<Element> <shells> End	Add GTO shells to AuxJK for element
NewAuxCGTO	<Element> "<AuxName>" <shells> End	Define new AuxC set for element via built-in name and/or custom shells
AddAuxCGTO	<Element> <shells> End	Add GTO shells to AuxC for element
NewCABSGTO	<Element> "<CABSName>" <shells> End	Define new CABS set for element via built-in name and/or custom shells
AddCABSGTO	<Element> <shells> End	Add GTO shells to CABS for element
NewECP	<Element> "<ECPName>" End	Define new built-in ECP for element
	<Element> <shells> End	Manually define new ECP for element (see Section 2.7.5)
DelECP	<Element>	Remove the ECP for the element
Setting basis sets for fragments (see Section 2.7.8)		
FragBasis	<FragID> "<BasisName>"	Define Basis for fragment
FragAuxJ	<FragID> "<AuxName>"	Define AuxJ for fragment
FragAuxJK	<FragID> "<AuxName>"	Define AuxJK for fragment
FragAuxC	<FragID> "<AuxName>"	Define AuxC for fragment
FragCABS	<FragID> "<CABSName>"	Define CABS for fragment
FragECP	<FragID> "<ECPName>"	Define ECP for fragment
ReadFragBasis	<FragID> "<filename.bas>"	Read Basis for fragment from file
ReadFragAuxJ	<FragID> "<filename.bas>"	Read AuxJ for fragment from file
ReadFragAuxJK	<FragID> "<filename.bas>"	Read AuxJK for fragment from file
ReadFragAuxC	<FragID> "<filename.bas>"	Read AuxC for fragment from file
ReadFragCABS	<FragID> "<filename.bas>"	Read CABS for fragment from file
ReadFragECP	<FragID> "<filename.bas>"	Read ECP for fragment from file
Reading basis sets from a file (see Section 2.7.9)		
GTOName	<filename.bas>	Read orbital basis from file
GTOAuxJName	<filename.bas>	Read AuxJ from file
GTOAuxName	<filename.bas>	Equivalent to GTOAuxJName
GTOAuxJKName	<filename.bas>	Read AuxJK from file
GTOAuxCName	<filename.bas>	Read AuxC from file
GTOCABSName	<filename.bas>	Read CABS from file
Removal of linear dependence (see Section 2.7.10)		
PCDTrimBas	Overlap	Trim the orbital basis in the overlap metric
PCDTrimAuxJ	Coulomb	Trim the AuxJ basis in the Coulomb metric
PCDTrimAuxJK	Coulomb	Trim the AuxJK basis in the Coulomb metric
PCDTrimAuxC	Coulomb	Trim the AuxC basis in the Coulomb metric
PCDThresh	-1	Threshold for the PCD (1e-16 to 1e-10 makes sense): chosen automatically if <0
AutoAux-related keywords (see Section 2.7.4)		
AutoAuxSize	0	Use minimal effective rather than minimal primitive exponent (suitable for ANO basis sets)
	1	(default) Increases the maximal exponent for the shells with low angular momenta.
	2	Increases the maximal exponent for all shells
	3	Directly uses the primitives and produces the largest fitting basis

continues on next page

Table 2.41 – continued from previous page

Keyword	Options	Description
AutoAuxLmax	false	If true, increase the maximal angular momentum of the fitting basis set to the highest value permitted by ORCA and by the orbital basis set.
AutoAuxLLimit	-1	If >0, do not exceed the given angular momentum.
AutoAuxF[0]	20.0	The factor to increase the maximal s-exponent
AutoAuxF[1]	7.0	Same for the p-shell
AutoAuxF[2]	4.0	Same for the d-shell
AutoAuxF[3]	4.0	Same for the f-shell
AutoAuxF[4]	3.5	Same for the g-shell
AutoAuxF[5]	2.5	Same for the h-shell
AutoAuxF[6]	2.0	Same for the i-shell
AutoAuxF[7]	2.0	Same for the j-shell
AutoAuxB[0]	1.8	Even-tempered expansion factor for the s-shell
AutoAuxB[1]	2.0	Same for the p-shell
AutoAuxB[2]	2.2	Same for the d-shell
AutoAuxB[3]	2.2	Same for the f-shell
AutoAuxB[4]	2.2	Same for the g-shell
AutoAuxB[5]	2.3	Same for the h-shell
AutoAuxB[6]	3.0	Same for the i-shell
AutoAuxB[7]	3.0	Same for the j-shell
AutoAuxTightB	true	Only use AutoAuxB[l] for shells with high l and AutoAuxB[0] for the rest
OldAutoAux	false	If true, selects the ORCA 3.1 generation procedure (deprecated)

2.8 Resolution-of-the-Identity (RI)

A very efficient and well-established way to speed up DFT calculations are resolution-of-the-idendity (RI) techniques also know as density-fitting.[111, 112, 113, 114, 115, 116, 117] Various variants to address the Coulomb part and Hartree-Fock exchange parts (cf. *Hartree-Fock* and *Hybrid DFT*). Available options available in ORCA include *RI-J*, *Split-RI-J*, *RIJONX*, *RI-JK*, and *RIJCOSX*.

Some Notes on RI in ORCA

- Any RI approximation requires the choice of a sufficiently large *auxiliary basis set*.
- *Split-RI-J* using the def2/J auxiliary basis is the default for *non-hybrid DFT*.
- *RIJCOSX* using the def2/J auxiliary basis is the default for *hybrid DFT*.
- When *scalar relativistic Hamiltonians* are used with all-electron basis sets, then SARC/J can be chosen as ageneral-purpose *auxiliary basis set*. Other choices are documented in the *basis sets section*.
- The usage of RI is generally recommended as the introduced error is very small. The speedup further enables the usage of larger *basis sets* for large systems which will improve the results significantly.
- The default usage of RI approximations can be disabled by the !NORI keyword (not recommended!).
- If you do not want to depend on the RI approximation, a reasonable yet inconvenient approach is to converge a RI-J calculation and then take the resulting orbitals as initial guess for a calculation with exact Coulomb term. This should converge within a few cycles and the total execution time should still be lower than just converging the calculation directly with exact Coulomb treatment.

2.8.1 RI-J

Note

This is the default for non-hybrid DFT! Can be turned off by using `!NORI`.

A very useful approximation that greatly speeds up DFT calculations unless the molecule gets very large is the so called "RI-approximation"

RI stands for "Resolution of the identity". In short, charge distributions arising from *products of basis functions* are approximated by a linear combination of *auxiliary basis functions*.

$$\phi_i(\vec{r})\,\phi_j(\vec{r}) \approx \sum_k c_k^{ij}\eta_k(\mathbf{r}) \tag{2.2}$$

There are a variety of different possibilities to determine the expansion coefficients c_k^{ij}. A while ago, Almlöf and coworkers [118] have shown that for the approximation of electron repulsion integrals, the best choice is to minimize the *residual repulsion*.

Note

The basic theory behind the RI method has been known for a long time and since at least the late sixties, methods similar to the RI approximation have been used — mainly in the context of "approximate ab initio methods" such as LEDO, PDDO, and MADO, but also in density functional theory in the mid and late seventies by Baerends, Dunlap, and others [111, 112, 113, 114]

Define:

$$R_{ij} \equiv \phi_i(\vec{r})\,\phi_j(\vec{r}) - \sum_k c_k^{ij}\eta_k(\vec{r}) \tag{2.3}$$

and

$$T_{ij} = \int\int R_{ij}(\vec{r})\,\frac{1}{|\vec{r}-\vec{r}'|}R_{ij}(\vec{r})\,d^3r d^3r' \tag{2.4}$$

Determining the coefficients that minimize T_{ij} leads to

$$\mathbf{c}^{ij} = \mathbf{V}^{-1}\mathbf{t}^{ij} \tag{2.5}$$

where:

$$t_k^{ij} = \left\langle \phi_i\phi_j \left| r_{12}^{-1} \right| \eta_k \right\rangle \tag{2.6}$$

$$V_{ij} = \left\langle \eta_i \left| r_{12}^{-1} \right| \eta_j \right\rangle \tag{2.7}$$

Thus, an ordinary two-electron integral becomes

$$\begin{aligned}\left\langle \phi_i\phi_j \left| r_{12}^{-1} \right| \phi_k\phi_l \right\rangle &\approx \sum_{p,q} c_p^{ij} c_q^{kl} V_{pq} \\ &= \sum_{p,q} V_{pq} \sum_r \left(\mathbf{V}^{-1}\right)_{pr} t_r^{ij} \sum_s \left(\mathbf{V}^{-1}\right)_{qs} t_s^{kl} \\ &= \sum_{r,s} \left(\mathbf{V}^{-1}\right)_{rs} t_r^{ij} t_s^{kl}\end{aligned} \tag{2.8}$$

and the total Coulomb energy becomes

$$
\begin{aligned}
E_J &= \sum_{i,j}\sum_{k,l} P_{ij}P_{kl}\left\langle \phi_i\phi_j \left| r_{12}^{-1} \right| \phi_k\phi_l \right\rangle \\
&\approx \sum_{i,j}\sum_{k,l} P_{ij}P_{kl}\sum_{r,s}\left(\mathbf{V}^{-1}\right)_{rs} t_r^{ij} t_s^{kl} \\
&= \sum_{r,s}\left(\mathbf{V}^{-1}\right)_{rs} \underbrace{\sum_{i,j} P_{ij} t_r^{ij}}_{\mathbf{X}_r} \underbrace{\sum_{k,l} P_{kl} t_s^{kl}}_{\mathbf{X}_s}
\end{aligned}
\tag{2.9}
$$

where **P** is the total density matrix.

In a similar way, the Coulomb contribution to the Kohn-Sham matrix is calculated. There are substantial advantages from this approximation: the quantities to be stored are the matrix $\mathbf{V}^{-1}$ — which depends only on two indices — and the three-index auxiliary integrals t_r^{ij}. This leads to a tremendous reduction of storage requirements compared to a four-index list of repulsion integrals. In addition, the two- and three-index electron repulsion integrals are easier to compute than the four-index integrals, leading to further reductions in processing time. Furthermore, the Coulomb energy and the Kohn-Sham matrix contributions can be quickly assembled by simple vector/matrix operations, leading to large time savings. This arises because each auxiliary basis function $\eta_k\left(\vec{r}\right)$ appears in the expansion of many charge distributions $\phi_i\left(\vec{r}\right)\phi_j\left(\vec{r}\right)$. Unfortunately, a similar strategy is less easily applied (or, at least, with less benefit) to the Hartree-Fock exchange term.

If the auxiliary basis set $\{\eta\}$ is large enough, the approximation is also highly accurate. Since any DFT procedure already has a certain, sometimes sizable, error from the noise in the numerical integration of the XC part, it might be argued that a similarly large error in the Coulomb part is perfectly acceptable without affecting the overall accuracy of the calculation much. Furthermore, the errors introduced by the RI method are usually much smaller than the errors in the calculation due to basis set incompleteness. It is therefore recommended to use the RI procedure for pure DFs. However, one should probably not directly mix absolute total energies obtained from RI and non-RI calculations as the error in the total energy accumulates and will rise with increased molecular size, while the errors in the relative energies will tend to cancel.

There are several choices for auxiliary basis sets described in the next section, which depend on the choice of the primary GTO basis set used to expand the molecular orbitals[1]. The RI procedure requires the inversion of the auxiliary basis metric **V**, which is often the most expensive matrix operation in the calculation, due to the O(N^3) scaling and the large size of the auxiliary basis. However, in ORCA this is done via an efficient Cholesky decomposition, which is only performed once during the startup phase of a single point caclulation. Hence, this step is usually of no concern in practice.

In ORCA, the RI approximation is toggled by the input

```
%method
  RI  on   # do use the RI-J approximation
      off  # do not use the RI-J approximation
end
```

Note

If you use RI, you *must* specify an auxiliary basis set (in the `%basis` section or using the appropriate simple keyword) or use the `!AutoAux` simple keyword.

[1] It probably should be noted that a slightly awkward step in the procedure is the inversion of the auxiliary integral matrix V, which can easily become very large. Matrix inversion is an $O(N^3)$ process such that for large molecules, this step takes some real time. However, in ORCA, this is only done once during the calculation, whereas other programs that constrain the fit to also exactly reproduce the number of electrons perform a similar process each iteration. Starting from ORCA 2.2.09, the Cholesky decomposition is used in favor of matrix inversion, removing any bottleneck concerning the solution of the linear equation system.

2.8.2 Split-RI-J

There is an improved version of the RI-algorithm that has been implemented since ORCA 2.2.09. This Split-RI-J algorithm yields the same Coulomb energy as the standard RI-algorithm, but is significantly faster if the basis set contains many high angular momentum functions (d-, f-, g-functions). For small basis sets, there is virtually no difference between the two algorithms, except that Split-RI-J uses more memory than standard RI. However, calculations with ca. 2000 basis functions only need about an extra 13 MB for Split-RI-J, which is a trivial requirement on present-day hardware.

The Split-RI-J algorithm is invoked with the `!Split-RI-J` simple keyword. Split-RI-J is presently only available for SCF and gradient calculations.

Note

- The Split-RI-J algorithm is the default if RI is turned on via `!RI`. If you do not want to use Split-RI-J, please also use the keyword `!NoSplit-RI-J`

2.8.3 RIJONX

Alternatively, the RI method can be used for the Coulomb term and the standard treatment for the exchange term. This method is called RIJONX since the exchange term should tend towards linear scaling for large molecules. RIJONX can be invoked via the `!RIJONX` keyword for Hartree-Fock and hybrid DFT calculations.

```
! HF RIJONX
```

The requirements for the auxiliary basis are the same as for the normal RI-J method.

2.8.4 RIJCOSX

Frustrated by the large difference in execution times between pure and hybrid functionals, we have been motivated to study approximations to the Hartree-Fock exchange term. The method that we have finally come up with is called the "chain of spheres" COSX approximation and may be thought of as a variant of the pseudo-spectral philosophy. Essentially, in performing two electron integrals, the first integration is done numerically on a grid and the second (involving the Coulomb singularity) is done analytically. For algorithmic and theoretical details see Refs. [119] and [120]. Upon combining this treatment with the *Split-RI-J* method for the Coulomb term (thus, a Coulomb fitting basis is needed!), we have designed the RIJCOSX approximation that can be used to accelerate Hartree-Fock and hybrid DFT calculations. Note that this introduces another grid on top of the DFT integration grid which is usually significantly smaller.

Note

Since ORCA 5, RIJCOSX is the default option for hybrid DFT (can be turned off by using `!NOCOSX`). However, it is by default NOT turned on for HF.

In particular for large and accurate basis sets, the speedups obtained in this way are very large - we have observed up to a factor of sixty! The procedure is essentially linear scaling such that large and accurate calculations become possible with high efficiency. The RIJCOSX approximation is basically available throughout the program. The default errors are on the order of 0.05 ± 0.1 kcal mol^{-1} or less in the total energies as well as in energy differences and can be made smaller with larger than the default grids or by running the final SCF cycle without this approximation. The impact on bond distances is a fraction of a pm, angles are better than a few tenth of a degree and soft dihedral angles are good to about 1 degree. To the limited extent to which it has been tested, vibrational frequencies are roughly good to 0.1 wavenumbers with the default settings.

The aim of this approximation is to efficiently compute the elements of exchange-type matrices as described in refs. [119, 120]:

$$K_{\mu\nu} = \sum_{\kappa\tau} P_{\kappa\tau}(\mu\kappa|\nu\tau) \tag{2.10}$$

where **P** is some kind of density-type matrix (not necessarily symmetric) and the two-electron integrals are defined over the basis set $\{\varphi\}$ by:

$$(\mu\kappa|\nu\tau) = \int \mu(\mathbf{r}_1)\kappa(\mathbf{r}_1)\nu(\mathbf{r}_2)\tau(\mathbf{r}_2)r_{12}^{-1}d\mathbf{r}_1 d\mathbf{r}_2 \tag{2.11}$$

The approximation pursued here can be written as follows:

$$K_{\mu\nu} \approx \sum_g X_{\mu g} \sum_\tau A_{\upsilon\tau}(\mathbf{r}_g) \sum_\kappa P_{\kappa\tau} X_{\kappa g} \tag{2.12}$$

Here, the index g refers to grid points $\mathbf{r}_g$ and:

$$X_{\kappa g} = w_g^{1/2}\kappa(\mathbf{r}_g) \tag{2.13}$$

$$A_{\upsilon\tau}(\mathbf{r}_g) = \int \frac{\nu(\mathbf{r})\tau(\mathbf{r})}{|\mathbf{r}-\mathbf{r}_g|}d\mathbf{r} \tag{2.14}$$

where w_g denotes the grid weights. Thus, the first integration is carried out numerically and the second one analytically. Note that this destroys the Hermitian character of the two-electron integrals.

Equation (2.12) is perhaps best evaluated in three steps:

$$F_{\tau g} = (\mathbf{PX})_{\tau g} \tag{2.15}$$

$$G_{\nu g} = \sum_\tau A_{\nu\tau}(\mathbf{r}_g)F_{\tau g} \tag{2.16}$$

$$K_{\mu\nu} = (\mathbf{XG}^+)_{\mu\nu} \tag{2.17}$$

As such, the equations are very similar to the pseudo-spectral method extensively developed and discussed by Friesner and co-workers since the mid 1980s and commercially available in the Jaguar quantum chemistry package. The main difference at this point is that instead of $X_{\kappa g}$ there appears a least-square fitting operator $Q_{\kappa g}$ in Friesner's formulation. Note that an analogue of the fitting procedure has also been implemented in ORCA and — in contrast to Friesner's pseudo-spectral method — does not need specially optimized grids. The basic idea is to remove the grid errors within the basis set by "fitting" the numerical overlap to the analytical one. Due to its nature, overlap fitting is supposed to work better with larger basis sets.

Basic Usage

The RIJCOSX approximation can be invoked by the `!RIJCOSX` keyword:

```
! HF RIJCOSX
```

By default, RIJCOSX will make use of the `def2/J` *auxiliary basis set*. Note that as the requirements for the SCF and correlation fitting bases are quite different an *correlation auxiliary basis set* has to be defined additionally. This is e.g. the case for *RI-MP2* or *double-hybrid DFT*.

```
! RI-MP2 def2-TZVPP def2/J def2-TZVPP/C RIJCOSX
```

> **Note**
>
> This is the default for hybrid DFT! The COSX part can be turned off by using the `!NOCOSX` keyword.

RIJCOSX Gradient

Given the exchange matrix, the exchange energy is given by (a sum over spin cases is left out here for simplicity):

$$E_{\mathrm{X}} = \frac{1}{2} \sum_{\mu\nu} P_{\mu\nu} K_{\mu\nu}(\mathbf{P}) \tag{2.18}$$

Previous to ORCA6, the gradient of the COSX contribution to the energy was taken as an approximation:

$$\frac{\partial E_X}{\partial \lambda} \approx 2 \sum_g \sum_{\mu\nu} \frac{\partial F_{\mu g}}{\partial \lambda} G_{\nu g} \tag{2.19}$$

with

$$\frac{\partial F_{\mu g}}{\partial \lambda} = w_g^{1/2} \sum_{\kappa} P_{\kappa\mu} \frac{\partial X_{\mu g}}{\partial \lambda} \tag{2.20}$$

as published in the original implementation paper [119].

Starting from ORCA6, this was updated to the full derivative of the energy component, including the derivative of all terms: grid weight derivatives, derivative of the SFitting matrices and all derivatives of F and G[2]. The gradient is thus now more accurate and less noisy. In case one wants to revert to the previous approximated version (not recommended), just set:

```
%method
  cosxgradtype grad_n
  useqgradfit false
end
```

COSX Numerical Grids

For expert users, the grid parameters for the exchange grids can be even more finely controlled:

```
%method
  IntAccX  Acc1, Acc2, Acc3
  GridX    Ang1, Ang2, Ang3
end
```

There are three grids involved: the smallest grid (Acc1, Ang1) that is used for the initial SCF iterations, the medium grid (Acc2, Ang2) that is used until the end of the SCF and the largest grid (Acc3, Ang3) that is used for the final energy and the gradient evaluations. `UseFinalGridX` can be used to turn this last grid on or off, though changing this is not generally recommended. More details about the grid constructions can be found in *Numerical Integration*.

SFitting Parameters

To modify the overlap fitting parameters, the following input can be specified:

```
%method
  UseSFitting false   # Same as NoSFitting in the simple input
                      # (Default is true)
  UseQGradFit false   # Uses the SCF fitting matrix for gradient calculations
                      # (Default is true)
end
```

Note that overlap fitting also works for HF and MP2 gradients without specifying any additional keyword. The `UseQGradFit` parameter uses the same fitting matrix for the gradients as for the energy calculation and is the default behavior since ORCA6.

[2] The theory is not yet published, but will be soon.

Partial Contraction Scheme

Since ORCA 5.0, generally-contracted basis sets can be handled efficiently by using an intermediate partially contracted (pc) atomic-orbital basis for the exchange-matrix computation without affecting the results [120]. Depending on the basis set and element type, computational speedups by many orders of magnitude are possible. For testing or benchmark purposes, the K matrix computation can be done in the original basis by using the flag

```
%method
  COSX_PartialContraction false   # No intermediate basis for generally contracted␣
↪shells
                                  # (Default is true)
end
```

Restoring Full Symmetry

The semi-numerical integration scheme in the default COSX algorithm breaks the permutational symmetry of the two-electron integrals. We have observed that this flaw is often the cause of convergence problems for iterative algorithms, in particular for multi-reference theories [121]. An input option is available since ORCA 6.0 to preserve the full eight-fold permutational symmetry of the two-electron integrals:

```
%method
  COSX_IntSymmetry Full      # Fully symmetrized integrals
                   Standard  # Original COSX algorithm
end
```

The full-symmetrization algorithm often improves the numerical stability and is enabled by default for TRAH-CASSCF and CASSCF linear-response calculations. However, the full-symmetrization algorithm may come with additional costs that depend on the number of **F** intermediates. The number of **F** intermediates depends on the symmetry of the density matrix (symmetric (S), anti-symmetric (A), and non-symmetric (N)) and whether overlap fitting is employed, as summarized in the table below.

Table 2.42: Number of COSX **F** intermediates per density matrix

S fitting	P symmetry (S, A, N)	Number of **F** intermediates
No	S / A	1
No	N	2
Yes	S / A	2
Yes	N	4

Note that for symmetric (S) and anti-symmetric (A) densities, we symmetrize and anti-symmetrize, respectively, exchange matrices at the end, which reduces the number of **F** intermediates by a factor of 2. The actual computational costs usually do not increase linearly with the number of **F** intermediates since we compute the costly analytic integrals ($A_{\nu\tau}(\mathbf{r}_g)$) only once and then contract them with the additional **F** intermediates. From our experience the overhead of the full-symmetrization algorithm is roughly between 1.2 and 1.5 times that of the original algorithm.

COSX Grid and Convergence Issues

Symptoms of convergence issues: Erratic convergence behavior, often starting from the first SCF step or possibly setting in at a later stage, which produce crazy energy values with "megahartree" jumps. If overlap fitting is on, the following error message may also be encountered: "Error in Cholesky inversion of numerical overlap".

Convergence issues may arise when the chosen grid has difficulties in representing the basis set. This is the "grid equivalent" of a linear dependence problem, as discussed in *Linear Dependence*. It should also be mentioned that the grid-related problem discussed here often goes hand in hand with basis set linear dependence, although not necessarily. The most straightforward way of dealing with this is to increase the size of the integration grid. This, however, is not always desirable or possible.

One way to avoid the Cholesky inversion issue is to turn overlap fitting off (`!NoSFitting`). However, this means that the numerical problems are still present, but are ignored. Due to the fact that overlap fitting operates with the numerical overlap and its inverse, it is more sensitive to linear dependence issues, so turning off the fitting procedure may lead to convergence. This may be a pragmatic — but by no means clean — solution, since it relies on the assumption that the numerical errors are small.

On the other hand, overlap fitting also gives a similar tool to deal with linear dependence issues as the one discussed for basis sets. The eigenvalues of the numerical overlap can be similarly inspected and small values screened out. There is unfortunately no universal way to determine this screening parameter, but see *Linear Dependence* for typical values.

The parameters influencing the method used for inversion and obtaining the fitting matrix are:

```
%method
  SFitInvertType Cholesky    # Cholesky inversion (default)
                 Cholesky_Q  # Cholesky + full Q matrix
                 Diag        # Inversion via diagonalization
                 Diag_Q      # Diag + full Q matrix
  SInvThresh     1e-8        # Inversion threshold for Diag and Diag_Q (default 1e-
↪8)
end
```

By default, the inversion procedure proceeds through Cholesky decomposition as it is the fastest option. Ideally, the overlap matrix is non-singular if the basis set is not linearly dependent. For singular matrices, the Cholesky procedure will fail. It should be noted at this point that the numerical overlap can become linearly dependent even if the overlap of basis functions is not, and so a separate parameter will be needed to take care of grid-related issues. To achieve this, a diagonalization procedure (`Diag`) can be used instead of Cholesky with the corresponding parameter to screen out eigenvectors belonging to eigenvalues below a threshold (`SInvThresh`). For both Cholesky and diagonalization procedures, a "full Q" approach is also available (`Cholesky_Q` and `Diag_Q`), which corresponds to the use of a more accurate (untruncated) fitting matrix.

2.8.5 RI-JK

An alternative algorithm for accelerating the HF exchange in hybrid DFT or HF calculations is RI-JK developed by Kendall and Früchtl[115]. It will use the RI approximation for both Coulomb and exchange and allows for a fair approximation to the Hartree-Fock exchange matrix. It can be difficult to make this approximation highly accurate. It is, however, usefully fast compared to direct SCF if the molecule is "dense" enough. There are special auxiliary basis sets for this purpose (see section *Basis Sets*). RI-JK is implemented in ORCA for SCF single point energies but not for gradients. The speedups for small molecules are better than for *RIJCOSX*, for medium sized molecules (e.g. $(\text{gly})_4$) similar, and for larger molecules RI-JK is less efficient than RIJCOSX. A detailed comparison can be found in Section 2.8.6. The errors of RI-JK are usually below 1 mEh and the error is very smooth (smoother than for *RIJCOSX*). Hence, for small calculations with large basis sets, RI-JK is a good idea, for large calculations on large molecules *RIJCOSX* is better.

Note

- RI-JK requires a larger auxiliary basis set. For the *Karlsruhe basis set*, the universal `def2/JK` and `def2/JKsmall` basis sets are available. They are large and accurate.
- For UHF RI-JK is roughly twice as expensive as for RHF. This is not true for RIJCOSX.
- RI-JK is available for conventional and direct runs and also for ANO bases. There the conventional mode is recommended.

Basic Usage

The RI-JK approximation can be invoked by the `!RI-JK` keyword:

```
! RHF def2/JK RI-JK
```

2.8.6 Speed Comparison of RIJCOSX and RI-JK

A comparison of the *RIJCOSX* and *RI-JK* methods is shown in Table 2.43.

Table 2.43: Comparison of wall clock times in seconds for a full SCF for $(gly)_2$, $(gly)_4$, and $(gly)_8$. Data taken from Ref. [122]

		def2-SVP	def2-TZVP(-df)	def2-TZVPP	def2-QZVPP
$(gly)_2$	*Default*	105	319	2574	27856
	RI-JK	44	71	326	3072
	RIJCOSX	70	122	527	3659
$(gly)_4$	*Default*	609	1917	13965	161047
	RI-JK	333	678	2746	30398
	RIJCOSX	281	569	2414	15383
$(gly)_8$	*Default*	3317	12505	82774	
	RI-JK	3431	5452	16586	117795
	RIJCOSX	1156	2219	8558	56505

It is obvious from the data that for small molecules the *RI-JK* approximation is the most efficient choice. For $(gly)_4$ this is already no longer obvious. For up to the def2-TZVPP basis set, *RI-JK* and *RIJCOSX* are almost identical and for def2-QZVPP *RIJCOSX* is already a factor of two faster than *RI-JK*. For large molecules like $(gly)_8$ with small basis sets *RI-JK* is not a big improvement but for large basis set it still beats the normal 4-index calculation. *RIJCOSX* on the other hand is consistently faster. It leads to speedups of around 10 for def2-TZVPP and up to 50-60 for def2-QZVPP. Here it outperforms *RI-JK* by, again, a factor of two.

2.8.7 Keywords

Table 2.44: Simple input keywords for the RI approximations.

Keyword	Description
`RI`	Activates *RI-J* (uses *Split-RI-J* by default)
`NORI`	Deactivates RI treaments
`SplitRIJ`, `Split-RI-J`	Activates *Split-RI-J*
`NoSplitRIJK`, `NoSplit-RI-J`	Activates *Split-RI-J*
`RIJONX`, `RIJDX`	Activates *RIJONX*
`RIJCOSX`, `RIJDX`	Activates *RIJCOSX*
`NORIJCOSX`, `NOCOSX`	Deactivates RIJCOSX, will activate RIJONX instead
`RIJK`, `RI-JK`	Activates *RI-JK*
`NoSFitting`	Deactivates *SFitting*

Table 2.45: `%method` block input keywords for the RI approximations.

Keyword	Options	Description
`RI`	`on/off`	Activates/deactivates RI (default: Split-RI-J)
`IntAccX`	`<Acc1, Acc2, Acc3>`	Control exchange grid accuracy parameters (cf. *Numerical Integration*)
`GridX`	`<Ang1, Ang2, Ang3>`	Controls exchange grid parameters (cf. *Numerical Integration*)
`UseSFitting`	`true/false`	Activates *SFitting* (default: true)
`SFitInvertType`	`Cholesky`	Uses Cholesky inversion (default)
	`Cholesky_Q`	Uses Cholesky + full Q matrix
	`Diag`	Activates inversion via diagonalization
	`Diag_Q`	Activates diagonalization + full Q matrix
`SInvThresh`	`<real>`	Inversion threshold for Diag and Diag_Q (default: 1e-8)
`UseQGradFit`	`true/false`	Activates SCF fitting matrix for gradient calculations (default: true)
`COSX_PartialContrac`	`true/false`	Activates intermediate basis for generally contracted shells (default: true)
`COSX_IntSymmetry`	`full`	Use fully symmetrized integrals
	`standard`	Use original COSX algorithm

2.9 Numerical Integration

Starting from its version 5.0, ORCA has a new scheme for the quadratures used in numerical integration. It is based on the same general ideas which were used for the old grids, except that we used machine learning methods, together with some final hands-on optimization, to find the optimal parameters for all atoms up to the 6th row of the periodic table, with the 7th row being extrapolated from that. For further details look at Ref. [120]. We also realized that the COSX and DFT grids have overall different requirements, and these were optimized separately.

The big advantage of this new scheme is that it is significantly more accurate and robust than the old one, even if having the same number of grid points. We tested energies, geometries, frequencies, excitation energies and properties to develop three new grid schemes named: DEFGRID1, DEFGRID2 and DEFGRID3, that will automatically fix all grids that are used in the calculations. DEFGRID1 behaves essentially like the old defaults, but it is more robust. The second is the new default, and is expected to yield sufficiently small errors for all kinds of applications. The last is a heavier, higher-quality grid, that is close to the limit if one considers an enormous grid as a reference.

In order to change from the default DEFGRID2, one just needs to add !DEFGRID1 or !DEFGRID3 to the main input.

It is also important to note that the COSX approximation is now the default for DFT, whenever HFexchange is neede. This can always be turned off by using !NOCOSX.

2.10 Details on the Numerical Integration Grids

As in all other popular grid schemes, our grids are constructed from assembling a set of atomic grids into a molecular one, using Becke's approach. Each individual atomic grid is build based on optimized parameters for that atom, and are composed of an angular and a radial part, that are defined separately.

The whole scheme was updated from ORCA 5.0, but we tried to keep things as close as possible to the previous one. First, the overall construction of these grids needs to be explained.

2.10.1 The Angular Grid Scheme

Instead of using a single angular grid throughout the whole atom, most schemes apply a so-called grid pruning in order to reduce the number of grid points outside of the most important regions, as we do in ORCA. In the current scheme, we split the atomic grids into five regions, using Lebedev grids with the following number of points on each of those:

Table 2.46: Different angular grid schemes used in ORCA. The numbers indicate the Lebedev grids used.

AngularGrid	Region 1	Region 2	Region 3	Region 4	Region 5
1	14	26	50	50	26
2	14	26	50	110	50
3	26	50	110	194	110
4	26	110	194	302	194
5	26	194	302	434	302
6	50	302	434	590	434
7	110	434	590	770	590

The ideal cutoffs between those regions were subjected to optimization, and are defined for all atoms. Whenever we refer to a AngularGrid flag in ORCA, one of these schemes is chosen.

2.10.2 The Radial Grid Scheme

The number of radial points (n_r) for a given atom is simply defined using the equation first defined by Krack and Köster:

$$n_r = (15 \times \varepsilon - 40) + b \times ROW$$

where ε is called the IntAcc of that grid in ORCA, b is any number and ROW refers to the row of the periodic table for that atom. In its original formulation, b was set to 5, but here it as now optimized and varies slightly depending on the AngularGrid schemes shown above.

One important thing to note is that each increase of IntAcc by 1, adds 15 radial points to the atomic grids, as in the previous grid scheme. These IntAcc values were optimized for each angular grid:

Table 2.47: Optmimized IntAcc parameters for the exchange-correlation and COSX grids.

AngularGrid	XC	COSX
1	4.004	3.816
2	4.004	4.020
3	4.159	4.338
4	4.388	4.871
5	4.629	4.871
6	4.959	4.871
7	4.959	4.871

After defining the number of radial points n_r, the actual radial grid is then defined from a Gauss-Chebyschev quadrature using the so-called M3 mapping from Treutler and Ahlrich:

$$r = \frac{\xi}{ln2} ln \frac{2}{1-x}$$

where $-1 \leq x \leq 1$, and the center of the grid ($x = 0$) coincides with the value of ξ. These ξ parameters were also optimized for each atom type.

2.10.3 The DEFGRIDs

With all that in mind, we can now present how the DEFGRIDs are built in terms of their AngularGrid scheme and IntAccs, which define the angular and radial parts of the atomic grids. More details can be found in Ref. [120].

Table 2.48: Angular grid schemes used in different part of ORCA. The XC and COSX grids are separated by a slash, and multiple COSX grid schemes are separated by a comma.

Grid Name	SCF	TD-DFT	CP-SCF	MP2 grad	MP2 2ndder
DEFGRID1	3 / 1, 1, 2	1 / 1	1 / 1	3	1
DEFGRID2	4 / 1, 2, 3	1 / 1	1 / 1	4	4
DEFGRID3	6 / 2, 3, 4	3 / 2	3 / 2	5	4

OBS.: The IntAccs for TD-DFT and the CP-SCF are 3.467 for the XC and 3.067 for the COSX instead of the default. These numbers can be smaller here and we exploit this to increase the overall speed.

From the Table 2.48 one can see, for instance, that the default SCF XC grid now is defined from AngularGrid 4 (with no extra final integration in the end). The default COSX uses a 1,2,3 grid scheme, with the COSX third grid being used to update the energy after the SCF converges and for the gradients.

2.10.4 Other Details and Options

The new adaptive pruning. The current pruning scheme uses lower grids close to the nucleus, and far away from the bonding region. However, if the basis set has polarized functions close to the nuclei, or diffuse Gaussians, this might not be sufficient.

To improve the grids for these problems, we now use by default an adaptive pruning scheme, that detects core-polarization, diffuse functions and steep basis set orbitals by analyzing the expectation value of the position operator, $\langle \hat{r} \rangle$, and fixes the grid accordingly. This can increase the grids in these cases by 10-20%, but gives significantly better results. To use the non-adaptive scheme, just set %METHOD GRIDPRUNING OLDPRUNING END. For a completely unpruned grid, set GRIDPRUNING to UNPRUNED.

A simpler Gauss-Legendre angular grid. By setting AngularGrid to 0, instead of using the Lebedev grids, a Gauss-Legendre angular grid will be built, as suggested by Treutler and Ahlrich [123]. The number of θ points is defined as $0.4n_r$ by default and the number of ϕ points is chosen as to avoid crowding close to the poles.

These grids are in general less efficient than the Lebedev's, but are useful if one needs to construct extremely large grids for specific applications.

The SpecialGrid Option. Sometimes, you might want to increase the integration accuracy for some atoms that need special care, while it is not necessary to enlarge the grid generally. ORCA provides you with a basic mechanism to increase the radial integration accuracy for a few atoms while maintaining the chosen grid for all others.

```
%METHOD
   # a maximum of 64 assignments can be made
   #  in = 0 : no changes are made
   #  in > 0 : the grid will be changed for all atoms with
   #           atomic number=in to IntAcc=an
   #  in < 0 : only the specific n'th atom will have its
   #           IntAcc value changed to an
```

(continues on next page)

(continued from previous page)

```
   SpecialGridAtoms  i1, i2, i3,...,in;
   SpecialGridIntAcc a1,a2,a3,...,an;
END
```

OBS.: Starting from ORCA 5.0 it is not necessary to use this option anymore unless you have very specific reasons. The basis set is considered during the grid construction and that is automatically extended if needed.

2.10.5 SCF Grid Keyword List

A complete description of the SCF grid options is given below. There are keywords specific to the XC integration, COSX integration and a general part that applies to all:

```
%METHOD
# XC grids
AngularGrid     1    #Lebedev50
                2    #Lebedev110
                3    #Lebedev194
                4    #Lebedev302
                5    #Lebedev434
                6    #Lebedev590
                7    #Lebedev770
                0    #SimpleGrid
IntAcc          5.0  # determines no. of radial points

# COSX grids
AngularGridX    1,1,2,4,5     # the first three are used in the SCF
                              # the 4th in the MP2 gradient and
                              # the 5th for MP2 second derivatives
IntAccX         3.56,-1,4.5   # if a -1 is given, the default IntAcc is used.

# General
NThetaMax       0.4           # only for AngularGrid=0, multiplier for nr
GridPruning     Unpruned      # no Pruning
                OldPruning    # the old pruning
                Adaptive      # default (and recommended)
HGridReduced    true          # Reduce grids for H and He by one
                              # unit (default and recommended)
BFCut           1e-10         # basis fcn. cut. Is adjusted according to
                              # convergence tolerances
WeightCut       1e-14         # grid weight cut. default: 1e-14
END
```

2.10.6 Changing TD-DFT, CP-SCF and Hessian Grids

TD-DFT. The grids used in CIS or TD-DFT can be changed in their respective block:

```
%TDDFT  # or %CIS, they are equivalent
  # XC grids
  IntAccXC 3.467
  GridXC   1

  #COSX grids
  IntAccX  3.076
  GridX    1
END
```

CP-SCF. The CP-SCF grids are changed in the %METHOD block:

```
%METHOD
  # XC grids
  Z_IntAccXC  3.467
  Z_GridXC    1

  #COSX grids
  Z_IntAccX   3.076
  Z_GridX     1
  Z_GridX_RHS 2
END
```

OBS.: The Z_Grid_RHS is only used in MP2 and the number here has a different meaning. It refers to which of the COSX grids used in the SCF will be chosen, rather than an AngularGrid scheme. The default is to use the second COSX grid.

Hessian. The XC grids used to compute the DFT terms in the Hessian are automatically chosen to be one unit higher than the SCF grids. Because of the second derivative terms, we found that it is better to have a slightly higher XC grid here. The COSX grid can be changed freely:

```
%FREQ
  HessGridX = 2,2,2,2
END
```

These four numbers refer to the possible usages of COSX in the Hessian, as explained in Sec. *Vibrational Frequencies*

Non-local functionals (VV10 and alike). The default non-local grid is defined by AngularGrid 2, and is not recommended to be changed. In any case, these can be altered by using:

```
%METHOD
  # non-local grids
  VdwAngularGrid    2        # same scheme as the SCF ones
  VdWIntAcc         5.0      # determines no. of radial points
  VdwGridPruning    Adaptive # default
  VdwDistTCut       10       # cutoff distance between grid points, in angstroem
END
```

2.10.7 When to Change from the Default Grids?

In general, the errors from the default grids are rather small and reasonable for most applications. After benchmarking against the GMTNK55 test set with the default !DEFGRID2, we found an error of about 0.01 ± 0.03 kcal/mol from DFT (compared to a reference grid), and 0.05 ± 0.10 kcal/mol for the COSX (compared to the analytical integration). We also benchmarked geometries, excitation energies and frequencies, and all errors are systematically low.

However, there might still be cases where an improved grid is needed:

- If you need to be confident that your energy error is below 0.2 kcal/mol;
- When dealing with anions with large negative charges (< -3);
- For very subtle intermolecular interactions;
- When dealing with weird electronic structures;
- With large conjugated systems - graphene-like structures and large polyaromatics.

When needed, the !DEFGRID3 is very large and conservative - it was built to cover almost all these cases. In contrast, !DEFGRID1 will yield grids of the size close to previous ORCA versions defaults, but still with increased accuracy.

2.11 Counterpoise Corrections

2.11.1 Boys-Bernardi Counterpoise Correction (BB-CP)

In calculating weak molecular interactions the nasty subject of the basis set superposition error (BSSE) arises. It consists of the fact that if one describes a dimer, the basis functions on A help to lower the energy of fragment B and vice versa. Thus, one obtains an energy that is biased towards the dimer formation due to basis set effects. Since this is unwanted, the Boys and Bernardi procedure aims to correct for this deficiency by estimating what the energies of the monomers would be if they had been calculated with the dimer basis set. This will stabilize the monomers relative to the dimers. The effect can be a quite sizable fraction of the interaction energy and should therefore be taken into account. The original Boys and Bernardi formula for the interaction energy between fragments A and B is:

$$\Delta E = E_{AB}^{AB}(AB) - E_A^A(A) - E_B^B(B) - \left[E_A^{AB}(AB) - E_A^{AB}(A) + E_B^{AB}(AB) - E_B^{AB}(B)\right] \tag{2.21}$$

Here $E_X^Y(Z)$ is the energy of fragment X calculated at the optimized geometry of fragment Y with the basis set of fragment Z. Thus, you need to do a total the following series of calculations:

1. optimize the geometry of the dimer and the monomers with some basis set Z. This gives you $E_{AB}^{AB}(AB)$, $E_A^A(A)$ and $E_B^B(B)$
2. delete fragment A (B) from the optimized structure of the dimer and re-run the single point calculation with basis set Z. This gives you $E_B^{AB}(B)$ and $E_A^{AB}(A)$.
3. Now, the final calculation consists of calculating the energies of A and B at the dimer geometry but with the dimer basis set. This gives you $E_A^{AB}(AB)$ and $E_B^{AB}(AB)$.

In order to achieve the last step efficiently, a special notation was put into ORCA which allows you to delete the electrons and nuclear charges that come with certain atoms but retain the assigned basis set. This trick consists of putting a ":" after the symbol of the atom. Here is an example of how to run such a calculation of the water dimer at the MP2 level (with frozen core):

```
#
# Calculations for Boys-Bernardi CP correction
#

# ---------------------------------------------
# First the monomer. In principle, you only need
# to run it once, but we keep it for clarity.
# ---------------------------------------------
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer"
* xyz 0 1
 O     7.405639     6.725069     7.710504
 H     7.029206     6.234628     8.442160
 H     8.247948     6.296600     7.554030
*

$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer"
* xyz 0 1
 O     7.405639     6.725069     7.710504
 H     7.029206     6.234628     8.442160
 H     8.247948     6.296600     7.554030
*

# ---------------------------------------------
# now the dimer
# ---------------------------------------------
$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
```

(continues on next page)

(continued from previous page)

```
%id "dimer"
* xyz 0 1
O   7.439917   6.726792   7.762120
O   5.752050   6.489306   5.407671
H   7.025510   6.226170   8.467436
H   8.274883   6.280259   7.609894
H   6.313507   6.644667   6.176902
H   5.522285   7.367132   5.103852
*

# ---------------------------------------------
# Now the calculations of the monomer at the
# dimer geometry
# ---------------------------------------------
$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer_1"

* xyz 0 1
O   7.439917   6.726792   7.762120
H   7.025510   6.226170   8.467436
H   8.274883   6.280259   7.609894
*

$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer_1"
* xyz 0 1
O   5.752050   6.489306   5.407671
H   6.313507   6.644667   6.176902
H   5.522285   7.367132   5.103852
*

# ---------------------------------------------
# Now the calculation of the monomer at the
# dimer geometry but with the dimer basis set
# ---------------------------------------------
$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer_2"
* xyz 0 1
O   7.439917   6.726792   7.762120
O : 5.752050   6.489306   5.407671
H   7.025510   6.226170   8.467436
H   8.274883   6.280259   7.609894
H : 6.313507   6.644667   6.176902
H : 5.522285   7.367132   5.103852
*

$new_job
! RHF MP2 cc-pVTZ VeryTightSCF PModel
%id "monomer_2"
* xyz 0 1
O : 7.439917   6.726792   7.762120
O   5.752050   6.489306   5.407671
H : 7.025510   6.226170   8.467436
H : 8.274883   6.280259   7.609894
H   6.313507   6.644667   6.176902
H   5.522285   7.367132   5.103852
*
```

From these calculations, you obtain the respective energies to compute the BB-CP correction:

Table 2.49: Energies calculated for the Boys-Bernardi counter-poise correction and final dimerization energies.

Component	E [a.u.]	E [kcal/mol]
$E_{AB}^{AB}(AB)$	-152.646980	
$E_{A}^{A}(A)$	-76.318651	
$E_{B}^{B}(B)$	-76.318651	
$E_{A}^{AB}(AB)$	-76.320799	
$E_{A}^{AB}(A)$	-76.318635	
$E_{B}^{AB}(AB)$	-76.319100	
$E_{B}^{AB}(B)$	-76.318605	
$\Delta E_{dim.}$	-0.009677	-6.07
ΔE_{BB-CP}	0.002659	1.67
$\Delta E_{dim.,corr.}$	-0.007018	-4.40

In this example, we calculated a BSSE correction of 1.67 kcal/mol.

It is also possible to set entire fragments as ghost atoms using the `GhostFrags` keyword as shown below. See section *Input of Coordinates* for different ways of defining fragments.

```
! MP2 cc-pVTZ VeryTightSCF PModel
* xyz 0 1
O   7.439917   6.726792   7.762120
O   5.752050   6.489306   5.407671
H   7.025510   6.226170   8.467436
H   8.274883   6.280259   7.609894
H   6.313507   6.644667   6.176902
H   5.522285   7.367132   5.103852
*
%geom
  GhostFrags {1} end # space-separated list and X:Y ranges accepted
  fragments
    1 {0 2 3} end
    2 {1 4 5} end
  end
end
```

Starting from ORCA 6.0, we support geometry optimizations with the counterpoise correction, using analytic gradients. This opens up the way of obtaining accurate non-covalent complex geometries (instead of just interaction energies) using modest basis sets. To use this functionality, one should NOT simply add `!Opt` to the above input files, but should instead use the dedicated compound script `BSSEOptimization.cmp` available in the ORCA Compound Script repository. Detailed usage are described in the comments of the compound script.

2.11.2 Geometrical Counterpoise Correction (gCP)

The central idea of the gCP correction [124] is to add a semi-empirical correction ΔE_{gCP} to the energies of molecular systems that removes artificial overbinding effects from BSSE. gCP uses atomic corrections and therefore also has the ability to correct for intramolecular BSSE. The parametrization is constructed such that it approximates the Boys and Bernadi counterpoise (CP) correction ΔE_{CP} in the intermolecular case. That is,

$$\Delta E_{CP} \approx \Delta E_{\text{gCP}} \tag{2.22}$$

where e.g. for a complexation reaction $A + B \rightarrow C$, our correction is given by

$$\Delta E_{\text{gCP}} = E_{\text{gCP}}(C) - E_{\text{gCP}}(A) - E_{\text{gCP}}(B) \tag{2.23}$$

In practice, E_{gCP} can be simply added to the HF/DFT energy

$$E_{\text{total}} = E_{\text{HF/DFT}} + E_{\text{gCP}} \tag{2.24}$$

which is done automatically in ORCA (the `FINAL SINGLE POINT ENERGY` includes the gCP correction).

The central equation of the gCP correction over all atoms N reads:

$$E_{\text{gCP}} = \sigma \cdot \sum_{a}^{N} \sum_{b \neq a}^{N} e_a^{\text{miss}} \cdot f_{\text{dec}}(R_{ab}) \tag{2.25}$$

where the energy e_a^{miss} is a measure of the incompleteness of the chosen target basis set (which is typically small) and $f_{\text{dec}}(R_{ab})$ is a decay function that depends on the interatomic distance R_{ab}. The scaling factor σ is one of 4 parameters needed for every `method/basis set` combination. More details on this can be found in the original gCP paper [124].

Analytical gradients with gCP are available for geometry optimization. Due to its semi-empirical nature, the correction is calculated within seconds, even for very large systems.

Basic Usage

The gCP correction can be invoked by the `!GCP(level)` keyword, where `level` is a combination of the method (Hartree-Fock, `HF` or density functional theory, `DFT`) and a basis set definition. See Table 2.50 for the available basis sets and the corresponding keywords. For example, gCP can be invoked in a B3LYP calculation with the def2-SV(P) basis set using the input:

```
! B3LYP def2-SV(P) GCP(DFT/SV(P))
```

The gCP input can also be defined in the `%method` block of the input:

```
! B3LYP def2-SV(P)

%method
  DoGCP        true
  GCPMETHOD    dft/sv(p)
end
```

Table 2.50: Overview of parametrized basis sets. The `level` keyword in `!GCP(level)` is a combination of `HF` or `DFT` and the `basis set` keyword. Valid inputs are, for example: `!GCP(HF/MINIS)`, `!GCP(DFT/LANL)`, `!GCP(HF/TZ)`, `!GCP(DFT/631GD)`, ...

Parametrized basis set	Definition	HF	DFT
MINIS	MINIS	yes	yes
SV	SV	yes	yes
6-31G(d)	631GD	yes	yes
6-31G(d) + LANL2DZ (Sc-Zn)	LANL	no	yes
def2-SV(P)	SV(P)	no	yes
def2-SVP	SVP	yes	yes
def2-TZVP	TZ	yes	yes

At all print levels, warnings from the gCP program are printed. Using the default print level, the only additional output is the final gCP correction before `FINAL SINGLE POINT ENERGY`. Using `!LargePrint` or `%output Print[P_gCP] 2 end` states the gCP `level`, the 4 parameters mentioned above, and the computed correction. A larger print level can be specified to get more details (more information on this below).

Several warnings and notices may be issued. Elements of the 5th and higher periods are treated as their 4th period analogs — e.g. Sn is treated with the same parameters as Ge. If this is the case, a note is printed. Another note is printed if there is a mismatch between the basis used for the SCF calculation and that of the requested gCP calculation. For example, the following input with tetramethyltin

```
! HF def2-SVP GCP(HF/MINIS)
*xyzfile 0 1 tetramethyltin.xyz
```

should use the parameters of Ge in place of Sn and there should be a mismatch between the basis set in ORCA (def2-SVP) and gCP (MINIS). Sure enough, the output is as follows:

```
 ** NOTE ** -> element sn will be treated as ge
NOTIFICATION: Different basis set in ORCA and otool_gcp:
ORCA: 142  gCP: 32
If you are NOT using ECPs, check your basis set inputs!
------------------   -----------------
gCP correction             0.073031339
------------------   -----------------
```

A mismatch between the basis sets used is allowed since a minor mismatch may only result in a small error. One should still be careful with such results; use your own judgment! This also allows gCP in calculations that use an unparametrized basis set. However, in this case, the number of basis functions and exponents should be very similar!

It should be noted that some elements are not parametrized, depending on the gCP `level` used. If only a few atoms in a large molecule are treated inaccurately or not at all, the error is expected to be small. To check all parameters used and the individual atomic contributions, specify the print level `%output Print[P_gCP] 3 end`. For example, the above tetramethyltin input with this print level has the following output:

```
-----------------------------------------------------------------------------

                  g C P  - geometrical counterpoise correction

-----------------------------------------------------------------------------
Method:  hf/minis

 ** NOTE ** -> element sn will be treated as ge
Parameters:   sigma        eta         alpha        beta
              0.1290      1.1526      1.1549       1.1763
Nbf:   32
NAtoms:   17

gCP element parameters:
  elem   emiss   nbas    elem   emiss   nbas    elem   emiss   nbas
    h    0.04240    1      he   0.02832    1      li   0.25266    2
    be   0.19720    2      b    0.22424    5      c    0.27995    5
    n    0.35791    5      o    0.47901    5      f    0.63852    5
    ne   0.83235    5      na   1.23292    6      mg   1.34339    6
    al   1.44828    9      si   1.61336    9      p    1.76814    9
    s    1.99201    9      cl   2.23311    9      ar   2.49323    9
    k    3.02964   10      ca   3.38998   10      sc   0.00000    0
    ti   0.00000    0      v    0.00000    0      cr   0.00000    0
    mn   0.00000    0      fe   0.00000    0      co   0.00000    0
    ni   0.00000    0      cu   0.00000    0      zn   0.00000    0
    ga   0.00000    0      ge   0.00000    0      as   0.00000    0
    se   0.00000    0      br   0.00000    0      kr   0.00000    0

      #     ON  sites    Nvirt           Emiss    BSSE [kcal/mol]
```

(continues on next page)

(continued from previous page)

```
      1      6     15      2.0          0.2799         10.0090
      2     32     16    -16.0          0.0000          0.0000
      3      6     15      2.0          0.2799         10.0084
      4      6     15      2.0          0.2799         10.0095
      5      6     15      2.0          0.2799         10.0083
      6      1     15      0.5          0.0424          0.4827
      7      1     15      0.5          0.0424          0.4828
      8      1     15      0.5          0.0424          0.4828
      9      1     15      0.5          0.0424          0.4827
     10      1     15      0.5          0.0424          0.4827
     11      1     15      0.5          0.0424          0.4827
     12      1     15      0.5          0.0424          0.4828
     13      1     15      0.5          0.0424          0.4827
     14      1     15      0.5          0.0424          0.4828
     15      1     15      0.5          0.0424          0.4827
     16      1     15      0.5          0.0424          0.4827
     17      1     15      0.5          0.0424          0.4827
Egcp:       0.0730313386 a.u.
NOTIFICATION: Different basis set in ORCA and otool_gcp:
ORCA: 142  gCP: 32
If you are NOT using ECPs, check your basis set inputs!
------------------   -----------------
gCP correction             0.073031339
------------------   -----------------
```

From this, it can be seen that the Sn atom (atom 2 in the list of atomic contributions) gives no contribution because its `Emiss` is zero (unparametrized for the given gCP `level`). This is confirmed by looking at the `gCP element parameters` section, which lists the `emiss` of Sn as zero for `!GCP(HF/MINIS)`. Rerunning this example with `!GCP(HF/SVP)` now gives a contribution for Sn, as seen by the following output. Note that this calculation also has a much smaller basis set mismatch, and so should be the more accurate gCP correction of the two.

```
-------------------------------------------------------------------------------

                  g C P  - geometrical counterpoise correction

-------------------------------------------------------------------------------
Method:  hf/svp

 ** NOTE ** -> element sn will be treated as ge
Parameters:   sigma        eta         alpha        beta
              0.2054      1.3157      0.8136       1.2572
Nbf:   148
NAtoms:   17

gCP element parameters:
  elem   emiss   nbas    elem   emiss   nbas    elem   emiss   nbas
    h    0.00811    5      he   0.00805    5      li   0.11358    9
    be   0.02837    9      b    0.04937   14      c    0.05538   14
    n    0.07278   14      o    0.10031   14      f    0.13327   14
    ne   0.17360   14      na   0.18114   15      mg   0.12556   18
    al   0.16719   18      si   0.14984   18      p    0.14540   18
    s    0.16431   18      cl   0.18299   18      ar   0.20567   18
    k    0.20096   24      ca   0.29966   24      sc   0.32599   31
    ti   0.30549   31      v    0.29172   31      cr   0.29380   31
    mn   0.29179   31      fe   0.29673   31      co   0.30460   31
    ni   0.24204   31      cu   0.35419   31      zn   0.35072   31
    ga   0.35002   32      ge   0.34578   32      as   0.34953   32
    se   0.36731   32      br   0.38201   32      kr   0.39971   32

      #     ON  sites    Nvirt           Emiss    BSSE [kcal/mol]
      1      6     16     11.0          0.0554          2.5093
```

(continues on next page)

(continued from previous page)

```
      2     32     16     16.0          0.3458          6.8274
      3      6     16     11.0          0.0554          2.5092
      4      6     16     11.0          0.0554          2.5094
      5      6     16     11.0          0.0554          2.5092
      6      1     16      4.5          0.0081          0.1703
      7      1     16      4.5          0.0081          0.1703
      8      1     16      4.5          0.0081          0.1703
      9      1     16      4.5          0.0081          0.1703
     10      1     16      4.5          0.0081          0.1703
     11      1     16      4.5          0.0081          0.1703
     12      1     16      4.5          0.0081          0.1703
     13      1     16      4.5          0.0081          0.1703
     14      1     16      4.5          0.0081          0.1703
     15      1     16      4.5          0.0081          0.1703
     16      1     16      4.5          0.0081          0.1703
     17      1     16      4.5          0.0081          0.1703
Egcp:       0.0301322002 a.u.
NOTIFICATION: Different basis set in ORCA and otool_gcp:
ORCA: 142  gCP: 148
If you are NOT using ECPs, check your basis set inputs!
------------------   -----------------
gCP correction             0.030132200
------------------   -----------------
```

General advice

- Small basis sets show not only a large BSSE, but general shortcomings. These effects are not always clearly distinguishable.
- If computationally affordable, large basis sets (triple-ζ and higher) are always preferable.
- gCP only makes sense for somewhat large molecules
- gCP should always be applied together with a dispersion correction. gCP in combination with D3 or D4 is well tested, but gCP with NL also works well (see chapter *Dispersion Corrections*).
- It has been demonstrated that the popular combination of B3LYP with 6-31G(d) can be strongly improved using DFT-D3 and gCP [125]. For convenience, the keyword `!B3LYP-gCP-D3/6-31G*` has been defined. This is equivalent to `!B3LYP 6-31G* D3BJ GCP(DFT/631GD)`.

Technical Notes

- `!GCP(HF/MINIS)` automatically sets the refitted D3 parameter, as proposed in the original gCP paper.
- The gCP method is implemented via an external tool called **otool_gcp**, which is based on the original Fortran program used in the publication. Thus, the `otool_gcp` binary can also be called directly via the command line (`otool_gcp -h` gives an overview of the options).
- It is also possible to read an external parameter file (`$HOME/.gcppar`) using the `!GCP(FILE)` keyword. For further information, please look at the manual for the gcp program as provided by the Grimme group (gCP manual).
- During the calculation, some temporary output files are written by ORCA: `BASENAME.gcp.in.tmp` and `BASENAME.gcp.out.tmp` will contain the input for `otool_gcp` and its output, respectively.

Keywords

Table 2.51: Simple input keywords for the gCP correction.

Keyword	Description
GCP(<level>)	Activates the gCP correction for a specified level of theory defined as <level>

Table 2.52: %method block input keywords and options for the gCP correction.

Keyword	Option	Description
DoGCP	true/false	Activate/deactivate the gCP correction
GCPMETHOD	<level>	Define the gCP level string, e.g. dft/svp
	hf/minis	Parameterization for HF with MINIS basis set
	hf/sv	Parameterization for HF with SV basis set
	hf/631gd	Parameterization for HF with 6-31G(d) basis set
	hf/svp	Parameterization for HF with def2-SVP basis set
	hf/tz	Parameterization for HF with def2-TZVP basis set
	dft/minis	Parameterization for DFT with MINIS basis set
	dft/sv	Parameterization for DFT with SV basis set
	dft/631gd	Parameterization for DFT with 6-31G(d) basis set
	dft/lanl	Parameterization for DFT with mixed 6-31G(d) + LANL2DZ (Sc-Zn) basis set
	dft/sv(p)	Parameterization for DFT with def2-SV(P) basis set
	dft/svp	Parameterization for DFT with def2-SVP basis set
	dft/tz	Parameterization for DFT with def2-TZVP basis set
	file	Read parameters from external parameterfile $HOME/.gcppar
GCP.D3MINIS	true/false	Use special DFT-D3 refit for HF/MINIS (default=true)

2.12 Relativistic Calculations

ORCA features three different approximations to cover relativistic effects:

1. The „Exact 2 component“ (*X2C*) Hamiltonian
2. The Douglas-Kroll-Hess (*DKH*) Hamiltonian to second order
3. The 0th order regular approximation (*ZORA*) with a model potential

Earlier versions of ORCA offered a number of additional approximations, which are no longer supported.

The main relativistic Hamiltonian that will be pursued in further development is the X2C Hamiltonian. Of the three alternatives, we believe that X2C has the best feature set and we recommend to all of our users to preferentially use this method.

All three relativistic model Hamiltonians are implemented for scalar relativistic energy calculations and these are carried through consistently through the entire program. Scalar relativity shows up as an additional effective potential that is added to the one-electron matrix. Scalar relativistic corrections to the two-electron interaction are not available in ORCA. Furthermore, self-consistent field calculations (HF, DFT, CASSCF) with inclusion of spin-orbit-coupling (SOC) are also not available in ORCA but we will not exclude the possibility to add this feature in a future version of the program.

2.12.1 Basic Usage

The relativistic methods in ORCA are implemented in a fairly straightforward way but do require some caution from the user. The basic options are available via simple input keywords and finer control is possible through the `%rel` block – see Section 2.12.9 for a full list of keywords. There are also settings specific to each Hamiltonian – these are explained in the respective section below.

The following three (mutually exclusive) input lines request, respectively:

- a TPSS optimization with the ZORA Hamiltonian, together with the *relativistically recontracted* `ZORA-def2-TZVP` basis set and the `SARC/J` *auxiliary basis set*;
- a DLPNO-CCSD(T) energy calculation with the second order DKH Hamiltonian with, together with a matching *correlation-consistent basis set* and suitable `AuxC`;
- a double-hybrid DFT NMR calculation with the X2C Hamiltonian and *appropriate basis sets*.

```
! TPSS Opt ZORA ZORA-def2-TZVP SARC/J
# or
! DLPNO-CCSD(T) DKH cc-pVTZ-DK cc-pVTZ/C
# or
! DSD-PBEP86/2013 NMR X2C x2c-TZVPall-s x2c/J
```

See Section 2.12.6 for an explanation of the basis set requirements in relativistic calculations and Section 2.7.3 for a list of available basis sets.

Warning

- Older versions of ORCA made extensive use of automatic basis set substitution and aliasing when the use of the DKH or ZORA Hamiltonians was detected. This is no longer the case! Relativistic versions of basis sets now have to be requested explicitly with the appropriate prefix. SARC basis sets also have to be requested explicitly for elements beyond Kr.
- Geometry optimizations with DKH and ZORA (but not X2C) automatically use the *one-center approximation*. When computing relative energies, do not mix energies from single-point calculations without the one-center approximation with those from geometry optimizations that do make use of this feature.

Scalar-Relativistic Gradients and Properties

Of the three model Hamiltonians, only X2C features analytic gradients. Hence, for geometry optimizations this is also the preferred method. For DKH and ZORA, the program automatically switches to the *one-center approximation*. This requires some attention by the users since final single point energies obtained with the one-center approximation are inconsistent with energies obtained without it. The one-center approximation is usually of sufficient accuracy but we have observed cases in actual applications where it leads to clearly wrong geometries. Hence, we strongly recommend to use the X2C Hamiltonian (or even an ECP-based calculation) in this realm.

If relativistic calculations are used for molecular properties there is a potential mismatch between non-relativistically calculated property integrals and the relativistic Hamiltonian. The procedure to remove these inconsistencies is referred to as "picture change". It is controlled by the keyword `PictureChange` in `%rel` and is activated by default in most calculations.

```
%rel
  FiniteNuc     true   # Invokes the Gaussian finite nucleus model.
  PictureChange 1 or 2 # First or second order picture change effects.
                       # Second order is potentially more accurate and more␣
↪expensive.
end
```

Picture change effects are implemented for DKH and X2C and to some extent also for ZORA. However, they are not implemented for all properties that ORCA can calculate. Please pay attention to the output of the property integral and property programs. Both programs will explicitly state which picture change effects are included in the molecular integrals. For more details, see Section 2.12.7. The example above also invokes the finite nucleus model – see Section 2.12.8.

2.12.2 Approximate Relativistic Hamiltonians

In the relativistic domain, calculations are based on the one-electron, stationary Dirac equation in atomic units (rest mass subtracted)

$$h_D\Psi = \left((\beta - 1)\, c^2 + c\boldsymbol{\alpha}\cdot\boldsymbol{p} + V\right)\Psi = E\Psi. \tag{2.26}$$

The spinor Ψ can be decomposed in its so-called large and small components

$$\Psi = \begin{pmatrix}\Psi_L\\ \Psi_S\end{pmatrix} \tag{2.27}$$

These are obviously coupled through the Dirac equation. More precisely, upon solving for Ψ_S, the following relation is obtained:

$$\Psi_S = \frac{1}{2c}\left(1 + \frac{E-V}{2c^2}\right)^{-1}\boldsymbol{\sigma}\cdot\boldsymbol{p}\Psi_L = R\Psi_L \tag{2.28}$$

Through the unitary transformation

$$U = \begin{pmatrix}\Omega_+ & -R^+\Omega\\ R\Omega_+ & \Omega_-\end{pmatrix}$$

with

$$\Omega_+ = \frac{1}{\sqrt{1+R^+R}}, \Omega_- = \frac{1}{\sqrt{1+RR^+}}$$

the Hamiltonian can be brought into block-diagonal form

$$U^+h_DU = \begin{pmatrix}\tilde{h}_{++} & 0\\ 0 & \tilde{h}_{--}\end{pmatrix} \tag{2.29}$$

The (electronic) large component thus has to satisfy the following relation

$$h_{++}\Psi_L = \Omega_+\left(h_{++} + h_\pm R + R^+\left(h_\mp + h_{--}R\right)\right)\Omega_+\Psi_L = E_+\Psi_L. \tag{2.30}$$

The approximate relativistic schemes implemented in ORCA (DKH and ZORA) use different methods to substitute the exact relation (2.30) with approximate ones, while the X2C formalism solves the model one-electron problem exactly.

One-Center Approximation

In the one-center approximation, which can be applied to any of the DKH, ZORA, or X2C methods, the model potential V in (2.26) only contains one-center terms, i.e. interactions with other atoms are ignored. This has several benefits:

- The unitary decoupling transformation (2.29) only needs to be solved for each atom type, within the atomic basis, not for the whole system.
- There are no gauge noninvariance errors in ZORA or IORA.
- Most importantly, there are no *picture-change effects* for geometric perturbations (i.e. gradients/Hessians), because the relativistic correction is independent of the molecular geometry. Thus, all relativistic methods can be used for geometry optimizations and frequency calculations at negligible computational effort, relative to the non-relativistic calculation.

Several papers in the literature show that this approximation is fairly accurate for the calculation of structural parameters and vibrational frequencies. Thus, it is enabled by default for DKH and ZORA calculations that require analytic gradients or Hessians, and indeed such calculations are not implemented without it (although fully numerical gradients/Hessians are available). For X2C, picture-change corrections for geometric perturbations are implemented, so the one-center approximation is not enabled by default. The approximation can be controlled explicitly via the simple keywords `!Rel1C` (on) and `!RelFull` (off), or the `OneCenter` keyword in the `%rel` block.

Warning

Energies obtained with and without the one-center approximation not comparable!

2.12.3 The Regular Approximation

In the regular approximation, (2.30) is approximated by

$$R = \frac{c}{2c^2 - V}\boldsymbol{\sigma}\cdot\boldsymbol{p}. \tag{2.31}$$

At the zeroth-order level (ZORA), $\Omega_\pm = 1$, so that the ZORA transformation is simply

$$U_{\text{ZORA}} = \begin{pmatrix} 1 & -R^+ \\ R & 1 \end{pmatrix} \tag{2.32}$$

and the corresponding Hamiltonian given by

$$\tilde{h}_{++}^{\text{ZORA}} = V + c\boldsymbol{\sigma}\cdot\boldsymbol{p}\frac{1}{2c^2 - V}c\boldsymbol{\sigma}\cdot\boldsymbol{p}. \tag{2.33}$$

At the infinite-order level (IORA), $\Omega_\pm$is taken into account, so that

$$U_{\text{IORA}} = U_{\text{ZORA}}\begin{pmatrix} \Omega_+ & 0 \\ 0 & \Omega_- \end{pmatrix} \tag{2.34}$$

and

$$\tilde{h}_{++}^{\text{IORA}} = \Omega_+\left(V + c\boldsymbol{\sigma}\cdot\boldsymbol{p}\frac{1}{2c^2 - V}c\boldsymbol{\sigma}\cdot\boldsymbol{p}\right)\Omega_+ \tag{2.35}$$

is the corresponding Hamiltonian. Note that despite the name – *infinite-order* regular approximation – this is still *not* exact.

In ORCA, the spin-free (scalar-relativistic) variant of ZORA and IORA are implemented. These are obtained from those above through the replacement

$$\boldsymbol{\sigma}\cdot\boldsymbol{p}\frac{1}{2c^2 - V}\boldsymbol{\sigma}\cdot\boldsymbol{p} \to \boldsymbol{p}\frac{1}{2c^2 - V}\boldsymbol{p}. \tag{2.36}$$

The regular Hamiltonians contain only part of the Darwin term and no mass-velocity term. A problem with relations (2.35) and (2.33) is that due to the non-linear dependence of the resulting regular Hamiltonians on V, a constant change of V, which in the Dirac and Schrödinger equations will result in a corresponding change of energy

$$E \to E + \text{const}$$

does not so in the regular approximation. Several attempts have been made to circumvent this problem. The scaled ZORA variant is one such procedure. Another one is given through the introduction of model potentials replacing V. Both approaches are available in ORCA.

The Scaled ZORA Variant

This variant goes back to van Lenthe et al. [126]. The central observation is that the Hamiltonian

$$h_{\text{scaledZORA}} = \frac{h_{\text{ZORA}}}{1 + \left\langle \Psi_L \left| c\boldsymbol{\sigma} \cdot \boldsymbol{p} \frac{1}{(2c^2 - V)^2} c\boldsymbol{\sigma} \cdot \boldsymbol{p} \right| \Psi_L \right\rangle} \tag{2.37}$$

produces constant energy-shifts $E \to E + \text{const}$ when the potential V is changed by a constant – *for hydrogenic ions*. For many-electron systems, the scaled-ZORA Hamiltonian still does not yield simple, constant energy shift for $V \to V + \text{const}$. But it produces the exact Dirac energy for hydrogen-like atoms and performs better than the first-order regular approximation for atomic ionization energies.

The Regular Approximation with Model Potential

The idea of this approach goes back to Van Wüllen [127], who suggested the procedure for DFT. However we also use it for other methods. The scalar relativistic ZORA self-consistent field equation is in our implementation (in atomic units):

$$\left[\mathbf{p} \frac{c^2}{2c^2 - V} \mathbf{p} + V_{\text{eff}} \right] \psi_i = \varepsilon_i \psi_i \tag{2.38}$$

where c is the speed of light. It looks like the normal nonrelativistic Kohn–Sham equation with the KS potential V_{eff}:

$$V_{\text{eff}}(\mathbf{r}) = -\sum_A \frac{Z_A}{|\mathbf{r} - \mathbf{R}_A|} + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + V_{\text{xc}}[\rho](\mathbf{r}) \tag{2.39}$$

(Z_A is the charge of nucleus A and R_A is its position; $\rho(r)$ is the total electron density and $V_{xc}[\rho]$ the exchange-correlation potential – the functional derivative of the exchange-correlation energy with respect to the density). The kinetic energy operator $T = -\frac{1}{2}\nabla^2$ of the nonrelativistic treatment is simply replaced by the ZORA kinetic energy operator:

$$T^{\text{ZORA}} = \mathbf{p} \frac{c^2}{2c^2 - V} \mathbf{p} \tag{2.40}$$

Clearly, in the regions where the potential V is small compared to c^2, this operator reduces to the nonrelativistic kinetic energy. V could be the actual KS potential. However, this would require to solve the ZORA equations in a special way which demands recalculation of the kinetic energy in every SCF cycle. This becomes expensive and is also undesirable since the ZORA method is not gauge invariant and one obtains fairly large errors from such a procedure unless special precaution is taken. Van Wüllen [127] has therefore argued that it is a reasonable approximation to replace the potential V with a model potential $\tilde{V}_{\text{model}}$ which is constructed as follows:

$$\tilde{V}_{\text{model}} = -\sum_A \frac{Z_A}{|\mathbf{r} - \mathbf{R}_A|} + \int \frac{\rho_{\text{model}}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + V_{\text{xc}}^{\text{LDA}}\left[\rho^{\text{model}}\right](\mathbf{r}) \tag{2.41}$$

The model density is constructed as a sum over spherically symmetric (neutral) atomic densities:

$$\rho_{\text{model}}(\mathbf{r}) = \sum_A \rho^A(\mathbf{r}) \tag{2.42}$$

Thus, this density neither has the correct number of electrons (for charged species) nor any spin polarization. Yet, in the regions close to the nucleus, where the relativistic effects matter, it is a reasonable approximation. The atomic density is expanded in a sum of s-type Gaussian functions like:

$$\rho^A(\mathbf{r}) = \sum_i d_i \exp\left(-\alpha_i \left|\mathbf{r} - \mathbf{R}_A\right|^2\right) \tag{2.43}$$

The fit coefficients were determined in three different ways by near basis set limit scalar relativistic atomic HF calculations and are stored as a library in the program. The fitting densities are available for elements up to Rn, as well as the actinoids. Through the variable `ModelDens` the user can choose between these fits and study the dependence of the results in this choice (it should be fairly small except, perhaps, with the heavier elements and the HF densities which are not recommended). The individual components of the model potential (eq. (2.41)) can be turned on or off through the use of the variable `ModelPot`. The available options for these keywords are listed in Section 2.12.9. For example:

```
! ZORA ZORA-def2-TZVP SARC/J
# for more detail use
%rel
     ModelPot 1,1,1,1
     ModelDens rhoZORA
end
```

Attention

The ZORA method is highly dependent on numerical integration and it is very important to pay attention to the subject of radial integration accuracy! By default, from ORCA 5.0 we consider that during the grid construction and the defaults should work very well. Only for very problematic cases, consider using a higher *`IntAcc` parameter* or at least to increase the radial integration accuracy around the heavy atoms using `SpecialGridAtoms` and `SpecialGridIntAcc` (see Section 2.10.4).

2.12.4 The Douglas-Kroll-Hess Method

The Douglas-Kroll-Hess (DKH) method expands the exact relation (2.30) in the external potential V. In ORCA the first- and second-order DKH methods are implemented. The first-order DKH Hamiltonian is given by

$$\tilde{h}^{(1)}_{++} = E_p + A_p V A_p + B_p V^{(p)} B_p, \tag{2.44}$$

with

$$E_P = \sqrt{c^4 + c^2 p^2},\ A_p = \sqrt{\frac{E_p + c^2}{2E_p}},\ B_p = \frac{c}{\sqrt{2E_p(E_p + c^2)}} \tag{2.45}$$

At second order, it reads

$$\tilde{h}^{(2)}_{++} = \tilde{h}^{(1)}_{++} + \frac{1}{2}\left[W_p, O\right] \tag{2.46}$$

where

$$\{W_p, E_p\} = \beta O,\ O = A_p\left[R_p, V\right]A_p,\ R_p = \frac{c\sigma p}{E_p + c^2} \tag{2.47}$$

define the second-order contribution. In ORCA, the spin-free part of $\tilde{h}^{(2)}_{++}$ is implemented.

The occurrence of the relativistic kinetic energy, E_P, which is not well-defined in position space, makes a transformation to the p^2-eigenspace necessary. Thus any DKH calculation will start with a decontraction of the basis set, to ensure a good resolution of the identity. Then the non-relativistic kinetic energy is diagonalized and the E_P-dependent operators calculated in that space. The potential V and $V^{(p)}$ are transformed to p^2-eigenspace. After all

contributions are multiplied to yield the (first- or second-order) Hamiltonian, the transformation back to AO space is carried out and the basis is recontracted.

The (spin-free) DKH-Hamiltonians contain all spin-free, relativistic correction terms, e.g. the mass-velocity and Darwin terms. As the potential enters linearly, no scaling or model potential is necessary to introduce the correct behaviour of the energy under a change

$$V \to V + \text{const.}$$

In all these respects the DKH Hamiltonians are much cleaner than the regular approximation Hamiltonians.

The first- or second-order DKH method be requested via the simple input keywords `DKH1` or `DKH2`, respectively (`DKH` is an alias for the latter), together with appropriate basis sets:

```
! DKH DKH-def2-TZVP SARC/J
```

For most calculations, no other settings are needed. A full list of available keywords is given in Section 2.12.9.

Magnetic Properties with DKH

For magnetic properties, the DKH transformation and consequently the DKH Hamiltonian and the corresponding property operators are not unique. Depending on whether the magnetic field is included in the free-particle Foldy–Wouthuysen (fpFW) transformation carried out in the first step of the DKH protocol or not, two different Hamiltonians result. If the magnetic field is included in the fpFW transformation, the resulting Hamiltonian is a function of the gauge invariant momentum

$$\boldsymbol{\pi} = \mathbf{p} + \mathbf{A}.$$

It is therefore gauge invariant under gauge transformations of the magnetic vector potential $\mathbf{A}$ and thus are the property operators derived from it. This is referred to as fπFW DKH Hamiltonian. If the magnetic field is not included in the FW transformation, the resulting Hamiltonian is a function of the kinetic momentum $\mathbf{p}$ only and thus is not gauge invariant. The latter Hamiltonian is referred to as fpFW DKH Hamiltonian. A comparison of both Hamiltonians is given in Table Table 2.53.

Table 2.53: Comparison of the properties of the fpFW and fπFW DKH Hamiltonians. For details see Ref. [128].

Criterion	fpFW Hamiltonian	fπFW Hamiltonian
Convergence of Eigenvalues to Dirac Eigenvalues	?	yes
1st order is bounded	no	yes
Reproduces Pauli Hamiltonian	no	yes
Gauge invariance	no	yes
Lorentz invariance	no	no

From this Table, it becomes clear that the fπFW DKH Hamiltonian is clearly preferred over the fpFW Hamiltonian. To obtain the property operators, it is however necessary to take the derivatives of these Hamiltonians. It turns out that in the case of the hyperfine-coupling tensor, the necessary derivatives produce divergent property operators in the case of the fπFW DKH Hamiltonian. This may be due to the unphysical assumption of a point-dipole as a source of the magnetic field of the nucleus. As a physical description of the magnetization distribution of the nucleus is not available due to a lack of experimental data, the magnetization distribution is assumed to be the same as the charge distribution of the nucleus, see Section *Finite Nucleus Model*. This is unphysical as the magnetization is caused by the one unpaired nucleon in the nucleus whereas the charge distribution is generated by the protons in the nucleus. So, physically, the magnetization should occupy a larger volume in space than the charge. This might also be the reason why the resulting finite-nucleus model is insufficient to remedy the divergences in the fπFW hyperfine-coupling tensor. Consequently, the hyperfine-coupling tensor is only implemented in the version resulting from the fpFW DKH Hamiltonian. In the case of the g-tensor both versions are implemented and accessible via the keyword

```
%rel fpFWtrafo true/false end
```

By default, this keyword is set to `true`. A detailed form of the property operators used for the g-tensor and hyperfine-tensors can be found in Ref. [128].

2.12.5 Exact Two-Component Theory (X2C)

The X2C implementation in ORCA closely follows that of Franzke, Weigend, and their coworkers, described in the references: [129] (energy), [130] (gradient), [131] (EPR hyperfine coupling), [132] (NMR spin–spin coupling), [133] (NMR shielding). These are also consistent with the work of Gauss and coworkers in refs: [134] (gradient, electric properties), [135] (Hessian), [136] (NMR shielding). However, despite the name, only a scalar relativistic (spin-free) version is available at present, resulting effectively in a one-component method (more aptly called "SF-X2C-1e"), very similar to the DKH and ZORA approaches described above. The main difference to the latter two is that the decoupling of the one-electron Dirac Hamiltonian is exact, rather than approximate. SF-X2C-1e is implemented in ORCA for energies, gradients, and various properties (see below) with both a point- and finite nucleus model.

We briefly describe the working equations here, using the notation of the aforementioned references, which differs somewhat from that in the previous sections. The one-electron Dirac equation is solved directly:

$$\mathbb{D}\mathbb{C} = \mathbb{M}\mathbb{C}\mathbb{E}, \quad \mathbb{C} = \begin{pmatrix} \mathbf{C}_+^{\mathrm{L}} & \mathbf{C}_-^{\mathrm{L}} \\ \mathbf{C}_+^{\mathrm{S}} & \mathbf{C}_-^{\mathrm{S}} \end{pmatrix} \tag{2.48}$$

to obtain the unitary transformation matrix:

$$\begin{aligned}
\mathbb{U} &= \begin{pmatrix} \mathbf{U}_{++} & \mathbf{U}_{+-} \\ \mathbf{U}_{-+} & \mathbf{U}_{--} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & -\mathbf{X}^\dagger \\ \mathbf{X} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}' \end{pmatrix} \\
\mathbf{X} &= \mathbf{C}_+^{\mathrm{S}} \left(\mathbf{C}_+^{\mathrm{L}}\right)^{-1} \\
\mathbf{R} &= \mathbf{S}^{-\frac{1}{2}} \left(\mathbf{S}^{-\frac{1}{2}} \tilde{\mathbf{S}} \mathbf{S}^{-\frac{1}{2}}\right)^{-\frac{1}{2}} \mathbf{S}^{\frac{1}{2}} \\
\tilde{\mathbf{S}} &= \mathbf{S} + \frac{1}{2c^2} \mathbf{X}^\dagger \mathbf{T} \mathbf{X}
\end{aligned}$$

which exactly block-diagonalizes the Hamiltonian:

$$\mathbb{U}^\dagger \mathbb{D} \mathbb{U} = \begin{pmatrix} \mathbf{h}^+ & \mathbf{0} \\ \mathbf{0} & \mathbf{h}^- \end{pmatrix} \tag{2.49}$$

$$\mathbf{h}^+ = \mathbf{R}^\dagger \mathbf{L} \mathbf{R} \tag{2.50}$$

$$\mathbf{L} = \mathbf{V} + \mathbf{T}\mathbf{X} + \mathbf{X}^\dagger \mathbf{T} + \mathbf{X}^\dagger \left(\frac{1}{4c^2}\mathbf{W} - \mathbf{T}\right) \mathbf{X} \tag{2.51}$$

where $\mathbf{V}$, $\mathbf{T}$, $\mathbf{S}$, and $\mathbf{W}$ are the potential, kinetic, overlap, and relativistic potential ($\hat{p}\,\hat{V}\hat{p}$) integral matrices. $\mathbf{h}^+$ is thus the matrix form of the relativistically-corrected one-electron Hamiltonian used for the rest of the calculation.

Note that the potential operator $\hat{V}$ used in the above equations only includes the electron–nuclear Coulomb interaction. External point charges may optionally be included via `%rel ModelPot[4]=1` (default 0). This option affects energy and NMR shielding calculations but it is presently not available for gradients or Hessians.

In the simplest case, it is sufficient to add the `X2C` simple keyword to the input and choose an appropriate basis set:

```
! X2C X2C-TZVPall X2C/J
```

DLU Approximation

The diagonal local approximation to the unitary transformation matrix (DLU), as introduced by Peng and Reiher,[137] reduces the computational cost of the X2C transformation by approximating $\mathbb{U}$ (i.e., $\mathbf{R}$ and $\mathbf{X}$) as an atomic-block-diagonal matrix. It leads to very minor loss of accuracy and is particularly suited for gradient/Hessian calculations.

$$\mathbf{R} \approx \bigoplus_A \mathbf{R}_A \quad \mathbf{X} \approx \bigoplus_A \mathbf{X}_A$$

where $\bigoplus_A$ denotes a direct sum of atomic diagonal blocks. Eqs (2.48)–(2.51) can then be solved independently for diagonal atomic blocks $\mathbf{h}_{AA}^+$. Off-diagonal blocks $\mathbf{h}_{AB}^+$ are obtained as:

$$\begin{aligned}
\mathbf{h}_{AB}^+ &= \mathbf{R}_A^\dagger \mathbf{L}_{AB} \mathbf{R}_B \\
\mathbf{L}_{AB} &= \mathbf{V}_{AB} + \mathbf{T}_{AB}\mathbf{X}_B + \mathbf{X}_A^\dagger \mathbf{T}_{AB} + \mathbf{X}_A^\dagger \left(\frac{1}{4c^2}\mathbf{W}_{AB} - \mathbf{T}_{AB}\right) \mathbf{X}_B
\end{aligned}$$

It is also possible to approximate light atoms a, b (below a certain atomic number) as non-relativistic:

$$\begin{aligned}
\mathbf{X}_a &= \mathbf{R}_a = \mathbf{I} \\
\mathbf{W}_{ab} &= \mathbf{W}_{aB} = \mathbf{0} \\
\mathbf{h}^+_{ab} &= \mathbf{V}_{ab} + \mathbf{T}_{ab} \\
\mathbf{h}^+_{aB} &= \left(\mathbf{V}_{aB} + \mathbf{T}_{aB}\mathbf{X}_B\right)\mathbf{R}_B
\end{aligned}$$

Unlike the one center approximation (which is also available for X2C), all nuclei are included in the potential operator $\hat{V}$.

The DLU approximation can be requested via the simple input keyword `!DLU-X2C`. Additional options are available in the `%rel` block (see also Section 2.12.9):

```
%rel
  DLU          false # default - not used
                true # turn DLU on
  LightAtomThresh 0 # (default) highest atomic number treated as non-relativistic
end
```

X2C Derivatives and Properties

As discussed in Section 2.12.7, when computing properties with relativistic methods, derivatives need to be taken of the correct Hamiltonian, namely $\mathbf{h}^+$ (eq (2.50)) in the X2C case. These include contributions due to the derivatives of $\mathbf{R}$ and $\mathbf{X}$, which are often small. Therefore, it is possible to neglect them and save some computational time via the following option:

```
%rel
  PictureChange 2 # compute the full relativistic Hamiltonian derivative
                1 # neglect derivatives of R and X
                0 # (default) use the non-relativistic operator
end
```

The same setting is applied to all properties for which the X2C correction is implemented. Currently, these are: geometric gradients, electric dipoles, quadrupoles, and polarizabilities, electric field gradients, EPR hyperfine couplings, and NMR shieldings and spin–spin couplings. For the Hessian, the X2C correction is implemented in a semi-numeric fashion. The DLU approximation is applied throughout, if requested, and reduces the computational effort dramatically. Note that for magnetic properties, the restricted magnetic balance (RMB) for the small component basis functions is used whenever GIAOs are requested (e.g. NMR shielding), while the restricted kinetic balance (RKB) is used otherwise (e.g. NMR coupling). The spin–orbit coupling integrals used for various properties only include a relativistic correction to the one-electron term, as is the case for DKH.

For second derivative properties whose number is proportional to the number of atoms, e.g. the DSO term of NMR couplings, first derivatives of various intermediate quantities required for the full X2C second derivative are stored on disk. The storage requirements can be reduced via the `StorageLevel` keyword and the missing intermediates will be recomputed on-the-fly, which of course increases the computation time.

```
%rel
  StorageLevel 0 # do not store anything - not always available
               1 # store integral derivatives
               2 # 1 + derivatives of R and X (default, minimum for DLU)
               3 # 2 + derivatives of L
               4 # 3 + derivatives of S-tilde
               5 # 4 + derivatives of D and M
end
```

2.12.6 Basis Sets in Relativistic Calculations

The different scalar relativistic potentials have different shapes in the core region. Consequently, each one of them requires specialized all electron basis sets that are optimized for the Hamiltonian at hand. Such basis sets have been designed, both as DKH and ZORA recontractions of the *Ahlrichs* and *Karlsruhe* basis sets for elements up to Kr (for which the original non-relativistic versions are all-electron), and as purpose-built segmented all-electron relativistically contracted (SARC) basis sets for elements beyond Kr .[50, 51, 52, 53, 54, 55] For X2C calculations, the *"x2c-XVPall" basis sets* and their variants are available.[57, 58] All available built-in choices are listed in the section *Relativistic Basis Sets*.

An uncontracted all-electron basis set of sufficient size will always work. Likewise, uncontracted fitting basis sets in all forms of RI calculations are always appropriate. Use the *`!Decontract` keyword* to decontract the chosen basis set and make it suitable for any relativistic Hamiltonian, as well as comparisons between them.

If large, uncontracted basis sets are used in scalar relativistic calculations, there is a distinct danger of variational collapse. This behavior is related to the fact that the relativistic orbitals will diverge for a point nucleus. ORCA features the Gaussian finite nucleus model of Dyall and Visscher for DKH and X2C. We recommend to always use this feature (`FiniteNuc`) in relativistic calculations (see Section 2.12.8).

Given the fact relativistic all-electron calculations on heavy element compounds feature very steep core basis functions, numeric integration, such as in DFT and COSX, may be challenging. ORCA features automatic procedures that adapt the integration grids for the presence of steep basis functions. However, in case you experience strange results, the numeric integration is one potential source of problem. The cure is to go to larger integration grids and, in particular, increase the *radial integration accuracy* (`IntAcc`).

2.12.7 Picture-Change Effects

Irrespective of which Hamiltonian has been used in the determination of the wave function, the calculation of properties requires some special care. This can be understood in two ways: First of all, we changed from the ordinary Schrödinger Hamiltonian to a more complicated Hamiltonian. As properties are defined as derivatives of the energy, it is clear that a new Hamiltonian will yield a new expression for the energy and thus a new and different expression for the property in question. Another way of seeing this is that through the transformation U, we changed not only the Hamiltonian but also the wave function. To obtain the property at hand as the expectation value of the property operator with the wave function, we have to make sure that property operator and wave function are actually given in the same space. This is done through a transformation of either the property operator or the wave function.

In any case, the difference between the non-relativistic and (quasi) relativistic property operator, evaluated over the (quasi) relativistic wave function, is called the *picture-change effect*. From what was said above, this is clearly not a physical effect. It describes how consistently the quasi relativistic calculation is carried out. A fully consistent calculation requires the determination of the wave function on the (quasi) relativistic level as well, as the use of the (quasi) relativistic property operator. This is obtained by enabling the `PictureChange` option in the `%rel` block:

```
%rel PictureChange 1 end  # or 2 - see below
```

It may be that the (quasi) relativistic and non-relativistic property operator do produce similar results. In this case, a calculation with picture-change turned off (`PictureChange=0`) may be a good approximation. This is often the case for valence shell-dominated properties, such as the eletric dipole and polarizability. On the other hand, property calculations involving operators that carry inverse powers of the electron-nucleus distance can be wildly inaccurate if picture change effects are ignored. It is therefore highly recommended to turn on picture-change in all (quasi) relativistic property calculations!

For DKH2, the fully consistent picture-change effects are obtained using the same transformation order for the property operator as for the one-electron Hamiltonian, i.e. setting `PictureChange=2`, while with `PictureChange=1` only first-order changes on the property operators are taken into account, which reduces the computational cost. However, since this is in no way a significant reduction, this choice is not recommended.

A similar argument applies to X2C, where `PictureChange=2` is the formally correct approach and `PictureChange=1` is an approximate treatment (see *X2C Derivatives and Properties*).

By default, ORCA tries to choose the most appropriate value of the `PictureChange` variable depending on the requested properties, as described below (and issues a warning at the start of the output), but explicit user input is

always respected.

- For geometric perturbations with DKH or ZORA the *one-center approximation* is activated and picture-change corrections are not needed. With X2C, `PictureChange=1` is chosen, because it significantly reduces the computational cost.
- For other properties, in particular core-sensitive ones (i.e. in EPR/NMR), the full picture-change correction is applied.
- Certain combinations of relativistic Hamiltonian and property operator are not implemented, in which case ORCA prints a warning, or aborts entirely if picture change effects are expected to be significant.

2.12.8 Finite Nucleus Model

Composite particles like nuclei have, as opposed to elementary particles, a certain spatial extent. While the point-charge approximation for nuclei is in general very good in nonrelativistic calculations, in relativistic calculations it might lead to non-negligible errors. A finite-nucleus model is implemented in ORCA – the Gaussian nucleus model of Ref. [138] – and is available for DKH and X2C (but not ZORA) calculations in the ORCA program package. It is accessible from the `%rel` block via

```
%rel FiniteNuc true end
```

By default, this keyword is set to `false`. If the keyword is set to `true`, finite-nucleus effects are considered in the following integrals:

- nucleus potential V and its nuclear derivatives;
- relativistic potential integral $V^{(p)}$ in DKH and X2C;
- one-electron part of the spin-orbit (SOC) integrals;
- electric-field gradient (EFG);
- contact density integral;
- Fermi-contact and spin-dipole terms of the EPR hyperfine coupling; and NMR indirect spin-spin coupling tensors;
- nucleus-orbit coupling integral;
- angular-momentum integral;
- NMR shielding integrals over GIAOs.

2.12.9 Keywords

Table 2.54: Simple input keywords for relativistic methods.

Keyword	Description
`DKH1`	Activate the first-order DKH Hamiltonian
`DKH2`, `DKH`	Activate the second-order DKH Hamiltonian
`ZORA`	Activate the ZORA Hamiltonian
`IORA`	Activate the IORA Hamiltonian
`X2C`	Activate the X2C Hamiltonian
`DLU-X2C`	Activate the X2C Hamiltonian in the DLU approximation
`RelDLU`	Enable the *DLU approximation* (X2C only)
`Rel1C`	Enable the *one-center approximation*
`RelFull`	Disable the one-center and DLU approximations

Table 2.55: `%rel` block input keywords relativistic methods.

Keyword	Options	Description
`Method`	`DKH`	Scalar relativistic Hamiltonian
	`ZORA`	
	`IORA`	
	`X2C`	
`ModelPot`	`<VeN>`, `<VC>`, `<VXa>`, `<VLDA>`, `<VPC>`	Flags for model potential (1: included, 0: not included). `<VeN>`: nuclear attraction; `<VC>`: Coulomb (ZORA/IORA only); `<VXa>`: Xalpha (ZORA/IORA only); `<VLDA>`: VWN-5 correlation (ZORA/IORA only); `<VPC>`: point charges (X2C only)
`ModelDens`	`rhoDKH`	Fitted atomic density used for Coulomb model potential and for PModel guess
	`rhoZORA`	
	`rhoHF`	
`OneCenter`	`false`	If true, use the *one-center approximation*
`C`	137.0359895	Speed of light used (in a.u.). Synonyms: `VELIT`, `VELOCITY`
`PictureChange`	0, 1, 2	*Picture change* for properties. 0: none; 1: enable; 2: full (DKH/X2C). Default: determine automatically
`FiniteNuc`	`false`	Whether to use the *finite nucleus model* (DKH or X2C)
`PrintLevel`	1	How much relativistics-related output to print (0-4). Alias for `%output Print[P_Rel]`
DKH-specific		
`Order`	2	Order of DKH Hamiltonian (1 or 2)
`fpFWtrafo`	`true`	Kind of Foldy-Wouthuysen transformation used for EPR with DKH (see Section 2.12.4). `true`: fpFW; `false`: fπFW
ZORA-specific		
`Xalpha`	0.7	Default value for Xalpha potential. Has effect only if VXa is included
`ScaleZORA`	`false`	Whether to scale the orbital energies (single points only)
X2C-specific		
`DLU`	`false`	If true, use the *DLU approximation* for X2C (mutually exclusive with `OneCenter`)
`LightAtomThresh`	0	Highest atomic number treated as non-relativistic in the *DLU approximation*
`StorageLevel`	2	X2C intermediate storage level (0–5) – see Section 2.12.5

2.13 Implicit Solvation

Implicit solvation models play an important role in quantum chemistry. Without resorting to placing multiple solvation shells of solvent molecules implicit solvent models are able to mimic the effect of a specific solvent on the solute.

The implicit solvent models available in ORCA are

1. C-PCM[139, 140] : The Conductor-like Continuum Polarization Model
2. SMD[141] : The Solvation Model based on Density
3. OpenCOSMO-RS[142] : Interface to the open source implementation of the COSMO-RS model
4. ALPB/ddCOSMO/CPCM-X : The solvation models available in XTB

Regarding C-PCM and SMD, they are natively implemented in ORCA. With these models, various types of calculations can be performed using a polarizable continuum with a realistic van der Waals cavity as summarized below:

- Energies of molecules in solution with a finite dielectric constant ε at the HF and DFT level, as well as for post-HF methods.
- Optimization of molecular structures in solution using HF or any DFT method with analytic gradients.
- Calculation of vibrational frequencies using the analytic Hessian for HF or any DFT method, provided that the same calculation is available in vacuum.
- Calculation of solvent effects on response properties like polarizabilities through coupled-perturbed SCF theory. For magnetic response properties, such as the g-tensor, the C-PCM response vanishes.
- Calculations of solvent shifts on transition energies using the time-dependent DFT or CIS method. The refractive index of the solvent needs to be provided in addition to the dielectric constant.
- First order perturbation estimate of solvent effects on state and transition energies in multireference perturbation and configuration-interaction calculations.

A detailed overview of the available implicit solvation methods and their usage is provided in Sections *The Conductor-like Polarizable Continuum Model (C-PCM)*, *The SMD Solvation Model*, *OpenCOSMO-RS* , *Extended Tight-Binding: GFN0-xTB, GFN-xTB, GFN2-xTB*, and *ONIOM Methods: General Overview*.

2.13.1 The Conductor-like Polarizable Continuum Model (C-PCM)

The conductor-like polarizable continuum model (C-PCM) is an implementation of the conductor-like apparent surface charge methods. In these models the solute is placed in a cavity of roughly molecular shape. The solvent reaction field is described by apparent polarization charges on the cavity surface, which are in turn determined by the solute. These charges can be treated as punctual (point charges) or be modelled as spherical Gaussians [143]. The cavity in ORCA is constructed differently depending on how the charges are treated. In the case of using point charges, the cavity is generated through the GEPOL[144, 145, 146] algorithm, either as solvent-excluding surface (SES), or solvent-accessible surface (SAS). When Gaussian charges are considered, the user can choose between a scaled vdW surface or the GEPOL SES, and the charge positions are determined following a Lebedev quadrature approach. This scheme is known as Gaussian Charge Scheme[140] and more details on how to use it are given in Section *Use of the Gaussian Charge Scheme*.

Within C-PCM the molecular Hamiltonian of the isolated system is perturbed by the solvent:

$$\hat{H} = \hat{H}^0 + \hat{V}$$

where $\hat{H}^0$ is the Hamiltonian of the isolated molecule, whereas $\hat{V}$ describes the solute – solvent interactions. The SCF procedure leads to the variational minimization of the free energy of the solute, G:

$$G = \left\langle \Psi \left| \hat{H}^0 \right| \Psi \right\rangle + \frac{1}{2} \left\langle \Psi \left| \hat{V} \right| \Psi \right\rangle$$

Using the conductor-like boundary condition the electrostatic potential can be determined by

$$V(\vec{r}) + \sum_i^{N_q} V_{q_i}(\vec{r}) = 0$$

where V and V_{q_i} are the electrostatic potential due to the solute and to the polarization charges, $\vec{r}$ is a point on the cavity surface, and N_q is the total number of solvation charges. The vector of polarization charge can then be determined by

$$\mathbf{AQ} = -\mathbf{V} \tag{2.52}$$

where the vector $\mathbf{V}$ contains the electrostatic potential due to the solute at the position of the charges. The elements of the matrix $\mathbf{A}$ have a different functional form depending on how the charges are treated. If we use point charges:

$$A_{ii} = 1.07\sqrt{\frac{4\pi}{S_i}}$$

$$A_{ij} = \frac{1}{r_{ij}} \tag{2.53}$$

in which S_i is the area of the surface element i, and $r_{ij} = |\vec{r}_i - \vec{r}_j|$. When Gaussian charges are considered:

$$A_{ii} = \frac{\zeta_i \sqrt{2/\pi}}{F_i}$$

$$A_{ij} = \frac{\text{erf}\left(\zeta_{ij} r_{ij}\right)}{r_{ij}}$$

Here, ζ_i is the exponent of the Gaussian charge i (i belongs to sphere I). This quantity is calculated as $\zeta_i = \zeta/(R_I\sqrt{w_i})$, where R_I is the radius of sphere I, w_i is the weight of the Lebedev point i, and ζ is a width parameter optimized for each particular Lebedev grid [143]. On the other hand, $\zeta_{ij} = \zeta_i\zeta_j/\sqrt{\zeta_i^2 + \zeta_j^2}$. The function

F_i, known as switching function, measures the contribution of the Gaussian charge i to the solvation energy. This function is calculated as

$$F_i = \prod_{J, i \notin J}^{\text{atoms}} g(\vec{r}_i, \vec{R}_J)$$

where $g(\vec{r}_i, \vec{R}_J)$ is the elementary switching function. In ORCA we use the improved Switching/Gaussian (ISWIG) function for $g(\vec{r}_i, \vec{R}_J)$ proposed in ref. [147]:

$$g(\vec{r}_i, \vec{R}_J) = 1 - \frac{1}{2}\left\{\text{erf}\left[\zeta_i\left(R_J - r_{iJ}\right)\right] + \text{erf}\left[\zeta_i\left(R_J + r_{iJ}\right)\right]\right\}$$

If $g(\vec{r}_i, \vec{R}_J) < 10^{-7}$ the value of g is set equal to 0.

If we consider a solvent with a dielectric constant ε, eq. (2.52) reads as

$$\mathbf{AQ} = -f(\varepsilon)\mathbf{V} \tag{2.54}$$

where $f(\varepsilon) = (\varepsilon - 1)/(\varepsilon + x)$ is a scaling function, and x is in the range 0-2. In C-PCM x is equal to 0.

The C-PCM model can be used via

```
! CPCM(solvent)
```

where `solvent` is one of the available solvents in Table 2.56

Table 2.56: List of available solvents for the C-PCM, SMD, openCOSMO-RS, and ALPB in ORCA. The data for the dielectric constant used within C-PCM is that at 293.15,[148] except for ammonia, which has a boiling point of 239.81 K. For the rest of solvation models, see the corresponding sources.[141][149] Notice that ALPB is only available in the context of XTB calculations. The list of solvents for ddCOSMO and CPCM-X is, on the other hand, provided in Table 3.24.

Solvent	C-PCM	SMD	COSMO-RS	ALPB
1,1,1-trichloroethane	X	X		
1,1,2-trichloroethane	X	X		
1,2,4-trimethylbenzene	X	X		
1,2-dibromoethane	X	X	X	
1,2-dichloroethane	X	X	X	
1,2-ethanediol	X	X		
1,4-dioxane / dioxane	X	X	X	X
1-bromo-2-methylpropane	X	X		
1-bromooctane / bromooctane	X	X	X	
1-bromopentane	X	X		
1-bromopropane	X	X		
1-butanol / butanol	X	X	X	
1-chlorohexane / chlorohexane	X	X	X	
1-chloropentane	X	X		
1-chloropropane	X	X		
1-decanol / decanol	X	X	X	
1-fluorooctane	X	X	X	
1-heptanol / heptanol	X	X	X	
1-hexanol / hexanol	X	X	X	
1-hexene	X	X		
1-hexyne	X	X		
1-iodobutane	X	X		
1-iodohexadecane / hexadecyliodide	X	X	X	
1-iodopentane	X	X		
1-iodopropane	X	X		

continues on next page

Table 2.56 – continued from previous page

Solvent	C-PCM	SMD	COSMO-RS	ALPB
1-nitropropane	X	X		
1-nonanol / nonanol	X	X	X	
1-octanol / octanol	X	X	X	X
1-pentanol / pentanol	X	X	X	
1-pentene	X	X		
1-propanol / propanol	X	X	X	
2,2,2-trifluoroethanol	X	X		
2,2,4-trimethylpentane / isooctane	X	X	X	
2,4-dimethylpentane	X	X		
2,4-dimethylpyridine	X	X		
2,6-dimethylpyridine	X	X	X	
2-bromopropane	X	X		
2-butanol / secbutanol	X	X	X	
2-chlorobutane	X	X		
2-heptanone	X	X		
2-hexanone	X	X		
2-methoxyethanol / methoxyethanol	X	X	X	
2-methyl-1-propanol / isobutanol	X	X	X	
2-methyl-2-propanol	X	X		
2-methylpentane	X	X		
2-methylpyridine / 2methylpyridine	X	X	X	
2-nitropropane	X	X		
2-octanone	X	X		
2-pentanone	X	X		
2-propanol / isopropanol	X	X	X	
2-propen-1-ol	X	X		
e-2-pentene	X	X		
3-methylpyridine	X	X		
3-pentanone	X	X		
4-heptanone	X	X		
4-methyl-2-pentanone / 4methyl2pentanone	X	X	X	
4-methylpyridine	X	X		
5-nonanone	X	X		
acetic acid / aceticacid	X	X	X	
acetone	X	X	X	X
acetonitrile / mecn / ch3cn	X	X	X	X
acetophenone	X	X	X	
ammonia	X		X	
aniline	X	X	X	X
anisole	X	X	X	
benzaldehyde	X	X	X	X
benzene	X	X	X	X
benzonitrile	X	X	X	
benzyl alcohol / benzylalcohol	X	X	X	
bromobenzene	X	X	X	
bromoethane	X	X	X	
bromoform	X	X	X	
butanal	X	X		
butanoic acid	X	X		
butanone	X	X	X	
butanonitrile	X	X		
butyl ethanoate / butyl acetate / butylacetate	X	X	X	
butylamine	X	X		
n-butylbenzene / butylbenzene	X	X	X	
sec-butylbenzene / secbutylbenzene	X	X	X	

continues on next page

Table 2.56 – continued from previous page

Solvent	C-PCM	SMD	COSMO-RS	ALPB
tert-butylbenzene / tbutylbenzene	X	X	X	
carbon disulfide / carbondisulfide / cs2	X	X	X	X
carbon tetrachloride / ccl4	X	X	X	
chlorobenzene	X	X	X	
chloroform / chcl3	X	X	X	X
a-chlorotoluene	X	X		
o-chlorotoluene	X	X		
conductor	X			
m-cresol / mcresol	X	X	X	
o-cresol	X	X		
cyclohexane	X	X	X	
cyclohexanone	X	X	X	
cyclopentane	X	X		
cyclopentanol	X	X		
cyclopentanone	X	X		
decalin	X	X	X	
cis-decalin	X	X		
n-decane / decane	X	X	X	
dibromomethane	X	X		
dibutylether	X	X	X	
o-dichlorobenzene / odichlorobenzene	X	X	X	
e-1,2-dichloroethene	X	X		
z-1,2-dichloroethene	X	X		
dichloromethane / ch2cl2 / dcm	X	X	X	X
diethyl ether / diethylether	X	X	X	X
diethyl sulfide	X	X		
diethylamine	X	X		
diiodomethane	X	X		
diisopropyl ether / diisopropylether	X	X	X	
cis-1,2-dimethylcyclohexane	X	X		
dimethyl disulfide	X	X		
n,n-dimethylacetamide / dimethylacetamide	X	X	X	
n,n-dimethylformamide / dimethylformamide / dmf	X	X	X	X
dimethylsulfoxide / dmso	X	X	X	X
diphenylether	X	X	X	
dipropylamine	X	X		
n-dodecane / dodecane	X	X	X	
ethanethiol	X	X		
ethanol	X	X	X	X
ethyl acetate / ethylacetate / ethanoate	X	X	X	X
ethyl methanoate	X	X		
ethyl phenyl ether / ethoxybenzene	X	X	X	
ethylbenzene	X	X	X	
fluorobenzene	X	X	X	
formamide	X	X		
formic acid	X	X		
furan / furane			X	X
n-heptane / heptane	X	X	X	
n-hexadecane / hexadecane	X	X	X	X
n-hexane / hexane	X	X	X	X
hexanoic acid	X	X		
iodobenzene	X	X	X	
iodoethane	X	X		
iodomethane	X	X		
isopropylbenzene	X	X	X	

continues on next page

Table 2.56 – continued from previous page

Solvent	C-PCM	SMD	COSMO-RS	ALPB
p-isopropyltoluene / isopropyltoluene	X	X		
mesitylene	X	X	X	
methanol	X	X	X	X
methyl benzoate	X	X		
methyl butanoate	X	X		
methyl ethanoate	X	X		
methyl methanoate	X	X		
methyl propanoate	X	X		
n-methylaniline	X	X		
methylcyclohexane	X	X		
n-methylformamide / methylformamide	X	X	X	
nitrobenzene / phno2	X	X	X	
nitroethane	X	X	X	
nitromethane / meno2	X	X	X	X
o-nitrotoluene / onitrotoluene	X	X		
n-nonane / nonane	X	X	X	
n-octane / octane	X	X	X	
n-pentadecane / pentadecane	X	X	X	
octanol(wet) / wetoctanol / woctanol				X
pentanal	X	X		
n-pentane / pentane	X	X	X	
pentanoic acid	X	X		
pentyl ethanoate	X	X		
pentylamine	X	X		
perfluorobenzene / hexafluorobenzene	X	X	X	
phenol	X		X	X
propanal	X	X		
propanoic acid	X	X		
propanonitrile	X	X		
propyl ethanoate	X	X		
propylamine	X	X		
pyridine	X	X	X	
tetrachloroethene / c2cl4	X	X	X	
tetrahydrofuran / thf	X	X	X	X
tetrahydrothiophene-s,s-dioxide /	X	X		
/ tetrahydrothiophenedioxide / sulfolane	X	X		
tetralin	X	X	X	
thiophene	X	X		
thiophenol	X	X		
toluene	X	X	X	X
trans-decalin	X	X		
tributylphosphate	X	X	X	
trichloroethene	X	X		
triethylamine	X	X	X	
n-undecane / undecane	X	X	X	
water / h2o	X	X	X	X
xylene	X	X		
m-xylene	X	X		
o-xylene	X	X		
p-xylene	X	X		

The parameters related to each implicit solvation method, as well as to the solvents can be more accurately defined using the `%cpcm` block input. A complete list of the available options are described in *Complete Keyword List for the %cpcm Block*.

The type of cavites surrounding the solute and how the solvation charges are treated in ORCA is explained in subsec-

tions *Use of the Gaussian Charge Scheme*, and *Use of the Point Charge Scheme*. ORCA supports two types of solvation charge schemes: (i) the point charge scheme, (ii) the Gaussian charge scheme. The default solvation scheme from ORCA 5.0 on is the Gaussian charge scheme with a vdW-type cavity (`surfacetype vdw_gaussian`). Older ORCA versions considered `surfacetype gepol_ses` and the point charge scheme as defaults. So, if one wants to reproduce the results obtained with ORCA versions older than ORCA 5.0, please use the tag `surfacetype gepol_ses` in the `%cpcm` block.

The different charge schemes are described in the subsections *Use of the Gaussian Charge Scheme*, and *Use of the Point Charge Scheme*. The availability of the analytical gradient and Hessian for the different combinations of charge scheme and surface is shown in Table Table 2.57.

Table 2.57: Available type of gradients and Hessians within the C-PCM in ORCA. The tag "YES" means that the feature is implemented, and "NO" that it isn't. For clarity, we denote by "*" the default scheme in ORCA (`surfacetype vdw_gaussian`).

`surfacetype`	Charge type	Gradient		Hessian	
		Analytical	Numerical	Analytical	Numerical
`gepol_ses`	Point	Yes	Yes	Yes	Yes
`gepol_sas`	Point	Yes	Yes	Yes	Yes
`vdw_gaussian^*`	Gaussian	Yes	Yes	Yes	Yes
`gepol_ses_gaussian`	Gaussian	No	Yes	No	Yes

Note 1: If the user wants to turn off the C-PCM, one has to write the following tag in the simple input:

```
! NOCPCM
```

This is needed, for instance, in the context of concatenated calculations using the `$new_job` feature, where the previous calculation involved solvation, but one wants to turn it off for the next one.

Note 2: As stated above, the default for C-PCM in ORCA is the Gaussian charge scheme with a vdW-type cavity. The implementation is done following ref. [140]. Hence, if C-PCM is used in ORCA calculations, one must cite this paper together with ref. [139]. Ref. [140] must also be cited if SMD, or openCOSMO-RS are requested, since these type of calculations rely on C-PCM.

Use of the Gaussian Charge Scheme

The Gaussian charge scheme avoids the Coulomb singularity present in conventional point charge surface element models. This approach, when applied together with a switching function, results in a smooth solvation potential and, more importantly, on smooth derivatives of this quantity with respect to external perturbations. Then, it is highly recommended to adopt this approach within the C-PCM. The Gaussian charge scheme can be used with two types of solute cavity surfaces: (1) a scaled vdW surface, (2) a solvent-excluded surface (SES). To assign the radii for the different atoms we follow the scheme proposed in ref. [147]. That is, we use Bondi radii [150] for all elements, except for hydrogen where we adopt 1.1 Å. For 16 of the main-group elements in the periodic table, where Bondi's radii are not defined, we adopt the radii proposed in ref. [151] by Mantina et al. This is the case for elements: Be, B, Al, Ca, Ge, Rb, Sr, Sb, Cs, Ba, Bi, Po, At, Rn, Fr, Ra. For the elements that are not covered neither by Bondi nor by Mantina, we consider a radius of 2 Å.

- **Scaled vdW cavity**

The Gaussian charge scheme with a scaled vdW-type cavity is now the default in ORCA, so one just needs to add the C-PCM tag in the `%cpcm` block in the input file. That would correspond to (although the user does not need to write that, as it is internally processed by ORCA):

```
%cpcm
  surfacetype vdw_gaussian
end
```

In this case, the radius R_I of atom I for the scaled vdW cavity is calculated as

$$R_I = f_{scal} R_I^{\text{vdW}}$$

where R_I^{vdW} is the vdW radius of atom I and f_{scal} is a scaling factor. This parameter is by default equal to 1.2, as suggested in ref. [147]. However, the user can modify its value through the `scale_gauss` tag in the `%cpcm` block in the input file. The number of C-PCM charges per atom is such that we have an approximate density of 5.0 charges/Å^2 on the surface of the cavity. This number corresponds approximately to 110 points on a hydrogen atom ($110/(4\pi \times 1.32^2) \approx 5.0\text{Å}^{-2}$). ORCA will choose then different levels of discretization (Lebedev grids) depending on the radii of the atoms. This scheme is called isodensity scheme. The threshold for the number of charges per unit of area on hydrogens and non-hydrogen atoms can be specified by the user via `thresh_h` and `thresh_noth`, respectively. Alternatively, the user can request a fixed number of Lebedev points per sphere in the cavity (independent of the radius of the sphere). This can be done via the `num_leb` tag. This parameter can adopt the following values: 50, 110, 194, 302, 434, 590, 770, 974, and 1202. ORCA versions older than ORCA 6.0 use this last scheme.

The analytical gradient, as well as the analytical Hessian are available for this solvation method.

- **Solvent-excluded surface**

The GEPOL-generated SES can be used together with the Gaussian charge scheme. In this case, the Gaussian charges are not only placed on the surface of the atomic spheres but also on the surface of the new spheres generated through the GEPOL algorithm. To use this approach we should modify the ORCA input file as follows

```
%cpcm
  surfacetype gepol_ses_gaussian
end
```

The SES is in general recommended when the solute is explicitly solvated by few solvation layers. In this case, the additional GEPOL spheres prevent the solvent to fill the space between explicit solvent molecules or between those molecules and the solute. The radius of the solvent sphere that rolls over the solute vdW-type surface to generate the SES is controlled by the keyword `rsolv`. This radius has a default value of 1.30 Å, but the user can change it by specifying another value for `rsolv` in the `%cpcm` block. The minimum radius for an added GEPOL sphere is controlled by `rmin`.

Neither the analytical gradient nor the analytical Hessian are available for this strategy. They should be computed numerically. Due to the interdependency between GEPOL spheres and the atoms present in our system, the analytical gradient computed using the SES does not converge as smooth as compared to that using the vdW-type cavity (see ref. [140]) and can lead to wrong minima. The Hessian is affected in the same way. Then, ORCA 6.0 does not support anymore the analytical gradient/Hessian for such surface to prevent inaccurate results.

Use of the Point Charge Scheme

Within this scheme, the solvation charges are treated as punctual and no switching function is used to accept/discard charges in the intersection between the spheres that form the solute cavity. These two facts lead to a discontinuous potential energy surface (numerical instabilities in the SCF energy and its derivatives). Then, the point charge scheme is not recommended within the C-PCM. If the user still wants to use this charge scheme it can be used together with two different type of surfaces for the solute cavity: (1) the SES, and (2) the SAS.

- **Solvent-excluded surface**

In the same way as for the Gaussian charge scheme with the SES, this surface is generated for the point charge scheme through the GEPOL algorithm. However, while for the case of `surfacetype gepol_ses_gaussian`, the surface is discretized using Lebedev-type grids, in the point charge scheme, the surface is divided in spherical triangles called "tesserae". The level of tesselation is controlled by the tag `ndiv`. The radii used for the solute atoms in order to construct the GEPOL cavity are those optimized for the COSMO-RS model[152] for H, C, N, O, F, S, Cl, Br, and I, while for the rest of elements scaled Bondi radii are used.

In order to use the point charge scheme with the SES, we should add the following tag in the `%cpcm` block:

```
%cpcm
  surfacetype gepol_ses
end
```

- **Solvent-accessible surface**

Here, the surface of the solute cavity is generated by following the center of a sphere with radius `rsolv` (representing the solvent molecule) rolling over the surface of the vdW surface of the solute. The discretization of the resulting surface is done via tesserae, as done for the SES. In order to use the point charge scheme together with the SAS, one should add the following tag in the `%cpcm` block:

```
%cpcm
  surfacetype gepol_sas
end
```

- **How to circumvent the numerical instabilities in the point charge scheme**

Numerical instabilities are implicit in the point charge scheme due to the "punctual" nature of the charges and to the fact that no switching function is considered in the intersection between the spheres that form the solute cavity. At the same time, there is no way to predict, in advance if there will be discontinuities in the SCF energy and/or in its derivatives. However, if the user still wants to use this charge scheme, no matter with which type of surface (SES or SAS), there are two things that one can try to minimize the aforementioned problems:

- **Change `ndiv`** : The parameter `ndiv` controls the number of triangles per sphere in the solute cavity. By changing the number of triangles, it also changes the number of triangles in the intersection between spheres and, then, this can solve those situations where two point charges get too close making the elements A_{ij} in eq (2.53) to diverge (and the solvation charges to suddenly have very large values). However, this strategy can be a solution for a particular case, but will never ensure that for the same system with another geometry the discontinuities in the SCF energy do not show (as it does not prevent point charges to be too close from each other).
- **Increase `pmin`** : This parameter removes those charges that are at a distance lower than `pmin` from each other. The default value for `pmin` is of 0.1 a.u. ($\approx$ 0.0529 Å). Then, by incrasing `pmin` one removes all pairs of charges that are too close from each other in the intersection between the spheres. **Note:** Be careful when increasing `pmin`. Although this prevents sudden jumps in the SCF energy, it can lead to "biasing" the solute-solvent interaction, as one is removing a significant number of charges that represent the effect of the solvent.

Calculation of the free energy of solvation within the C-PCM

The solvation free energy, ΔG_{solv}, is defined as the free energy of transfer of a solute from the gas phase to the condensed phase. This quantity can be written as (see eq 27 in ref. [140])

$$\Delta G_{solv} = E_{solv}(\vec{R}_l, \vec{R}_v) + \Delta G_{el} + \Delta G_{cav} + \Delta G_{disp} \tag{2.55}$$

The first term, $E_{solv}(\vec{R}_v, \vec{R}_l)$, corresponds to the difference between the liquid-phase expectation value of the gas-phase Hamiltonian, $E(\vec{R}_l)$, and the gas-phase potential energy surface $E(\vec{R}_v)$

$$E_{solv}(\vec{R}_v, \vec{R}_l) = E(\vec{R}_l) - E(\vec{R}_v)$$

The second term in eq (2.55) accounts for the electronic-polarization contribution to ΔG_{solv} and is calculated from the charges spread over the surface of the solute cavity (vdW-type or the SES). Finally, ΔG_{cav} is the cavitation energy, that is, the reversible work required to create a cavity inside the bulk of the solvent in order to accommodate the solute, while ΔG_{disp} accounts for the changes in the dispersion energy occuring when solvating the solute. The sum of these last two terms correspond to the non-electrostatic contribution, ΔG_{nel}, to ΔG_{solv}

$$\Delta G_{nel} = \Delta G_{cav} + \Delta G_{disp}$$

In ORCA, if a system is solvated within the C-PCM (defining `!CPCM(solvent name)` in the input file, within either the point charge scheme or the Gaussian charge scheme) there is no "non-electrostatic solvation contribution"

neither to the SCF energy, nor to its derivatives. In order to have a rough idea of the value of ΔG_{nel}, one can estimate this quantity through empirical equations available in the literature obtained from experimental data. For instance, ΔG_{nel} can be estimated from the free energy of hydration for linear-chain alkanes[153]

$$\Delta G_{\text{nel}} = 1.321 + 0.0067639 \times \text{SASA} \tag{2.56}$$

where SASA is the solvent-accessible surface area. However, although eq (2.56) may give a good estimation of ΔG_{nel} for organic molecules in water, it is a bad approximation for non-electrostatic solvation effects when considering solvents different than water (as well as for many solutes in water). There are two alternatives to this approximation that one can adopt in order to include non-electrostatic solvation effects in ORCA calculations:

1. The use of the SMD model in combination with the C-PCM (see section *The SMD Solvation Model*). This is the most adopted strategy within the QM community.
2. The use of Gaussian charges to compute the electrostatic contribution to ΔG_{solv} together with an equation for ΔG_{nel} calculated from the contribution of the solute atoms to the SASA.[140]

Focusing on the second strategy, the cavitation energy, ΔG_{cav} is calculated through the equation proposed by Pierotti and Claverie.[154, 155] In this case, ΔG_{cav} is expanded in powers of the ratio $\overline{\mathbf{R}}$ between the radius of the solute sphere R_I and that of the solvent spheres R_S. Then, the cavitation energy for the solute sphere centered at atom I reads as,

$$\Delta G_{cav,I} = -\ln(1-y) + \left(\frac{3y}{1-y}\right)\overline{\mathbf{R}} + \left[\frac{3y}{1-y} + \frac{9}{2}\left(\frac{y}{1-y}\right)^2\right]\overline{\mathbf{R}}^2 + \frac{yP}{\rho_S k_B T}\overline{\mathbf{R}}^3$$

with $y = 8\pi\rho_S R_S^3/6$, being ρ_S the density of the solvent, $\overline{\mathbf{R}} = R_I/R_S$, T the temperature and P the pressure. If we consider the solvent-accessible surface (SAS), then, the total ΔG_{cav} is the sum of the $\Delta G_{cav,I}$ weighted by a factor that depends on the exposed SASA_I of sphere I

$$\Delta G_{cav} = \sum_{i=1}^{N_{\text{sph}}} \frac{\text{SASA}_I}{4\pi R_I^2}\Delta G_{cav,I} = \sum_{i=1}^{N_{el}} \frac{\text{SASA}_i}{4\pi R_I^2}\Delta G_{cav,i} \tag{2.57}$$

Here, the summation over the total number of spheres (N_{sph}) that conform the SAS is replaced by a summation over the total number of surface elements (N_{el}) in which the SAS is divided into.

With respect to ΔG_{disp}, this quantity is assumed to depend linearly on the contribution of each surface element to the SASA

$$\Delta G_{disp} = \sum_{i=1}^{N_{\text{el}}} \sigma_I \text{SASA}_i \tag{2.58}$$

Here, the factor σ_I is the atomic surface tension of the sphere I to which the surface element i belongs to.

This strategy, that considers eqs (2.57) and (2.58) to compute ΔG_{nel} assumes the use of Gaussian charges for the calculation of ΔG_{el} in eq (2.55). When a vdW-type cavity is used to account for electrostatic solvation effects, then the solvation approach is called GVDW_nel, while when a SES-type cavity is used we refer to the solvation approach as GSES_nel ("G" for Gaussian charges and "nel" because of the inclusion of non-electrostatic solvation effects). Details on the GVDW_nel and GSES_nel models can be found in ref. [140].

The GVDW_nel and GSES_nel models have been parametrized for solutes containing H, C, N and O atoms in the following solvents: benzene, chloroform, cyclohexane, octanol, toluene and water. The parameters for the σ_I atomic surface tensions in eq (2.58) together with the radius of the solvent molecules R_S used to generate the SAS are provided in ref. [140]. The use of the GVDW_nel and GSES_nel is controlled by the string `cds_cpcm` to add in the `%cpcm` block. This flag should be equal to 2 within these two models (by default, when we use C-PCM together with the Gaussian Charge Scheme without non-electrostatic solvation effects, `cds_cpcm` is equal to 0). In order to calculate ΔG_{solv} for the GVDW_nel scheme one has to do the following steps:

1. Optimize the geometry of the solute (anisole in this case) in vacuum. For a DFT calculation at the B3LYP/def2-TZVP level of theory (with RIJCOSX), our input file would read like:

```
! B3LYP def2-TZVP D3BJ tightopt

%maxcore 4000

* xyz 0 1
C   0.66588156410505    1.19833878437922    -0.02647672759883
C   1.95470174000504    0.69483871574862    -0.01927812843331
C   2.17893173780041   -0.68097630146538     0.00933914777704
C   1.09270919891774   -1.54252176637089     0.03141990372830
C  -0.21073409652496   -1.05179390481975     0.02471602253843
C  -0.42517201252444    0.32533458015547    -0.00499572635290
H   0.47565631823749    2.26336213776820    -0.05012843684387
H   2.79210971512409    1.38085300778245    -0.03749594945361
H   3.18771596563920   -1.07149825338966     0.01433233618265
H   1.25061585302875   -2.61357041387417     0.05495238973649
H  -1.03877793934640   -1.74453554370491     0.04346162651229
O  -1.65366310927299    0.90979347913477    -0.01693978807213
C  -2.79791686738707    0.07370821312998     0.01117694976431
H  -2.83142307710305   -0.58944196707467    -0.85867919998175
H  -3.65619850989594    0.74176224021803    -0.00958531946332
H  -2.82883648080292   -0.53125300761732     0.92288089996024
*
```

2. Use the resulting structure (`anisole.xyz`) as input structure for the subsequent geometry optimization in solution. In this case, we consider water as the solvent. The input file looks like:

```
! B3LYP def2-TZVP D3BJ tightopt CPCM(Water)

%cpcm
  cds_cpcm 2
end

%maxcore 4000

* xyzfile 0 1 anisole.xyz
```

3. Search for the `FINAL SINGLE POINT ENERGY` in the output file for the solvated system (we call it E_s):

```
-------------------------   --------------------
FINAL SINGLE POINT ENERGY      -346.723542928703
-------------------------   --------------------
```

4. Search for the `FINAL SINGLE POINT ENERGY` in the output file for the system in vacuum (we call it E_0):

```
-------------------------   --------------------
FINAL SINGLE POINT ENERGY      -346.719657761697
-------------------------   --------------------
```

5. Calculate $\Delta G_{solv} = E_s - E_0 = -0.003885$ a.u. If we convert it to kcal/mol and round it to two decimal digits, we have $\Delta G_{solv} = -2.44$ kcal/mol. This quantity is in very good agreement with its experimental counterpart (-2.45 kcal/mol[141]).

6. In case the user is interested in the value of ΔG_{nel}, this quantity is printed in the `CPCM Solvation Model Properties` block:

```
CPCM Solvation Model Properties:
Surface-charge          :  -0.02580688416088
Corrected charge        :   0.00000000000000
Outlying charge corr.   :   0.00005046206545 Eh    0.00137 eV
Free-energy (cav+disp)  :   0.00272597402638 Eh    0.07418 eV
```

Here, ΔG_{nel} corresponds to `Free-energy (cav+disp)` and is calculated through eqs (2.57) and (2.58). The term called `Surface-charge` corresponds to the solute net charge that lies outside the C-PCM cavity. It is basically the sum of the C-PCM charges (without the scaling by $f(\varepsilon)$). The effect of this excess of charge on the energy and the C-PCM charges themselves can be corrected by the so-called outlying charge correction. In ORCA, this correction is calculated through a Lagrangian-based algorithm,[156] but only printed for information purposes. That is, the outlying charge effect is neither added to the SCF energy nor to its derivatives. In any case, the corrected total charge is printed in `Corrected charge`, while the correction term for the energy is printed in `Outlying charge corr.`. To further help the user, ORCA also prints a file with extension `.cpcm_corr`, where the corrected C-PCM charges are provided.

If we want to use, instead, the GSES_nel model, one just needs to add `surfacetype gepol_ses_gaussian` in the `%cpcm` block.

Note: The analytical Hessian is not available for the GVDW_nel and GSES_nel models as the second derivative of ΔG_{nel} with respect to nuclear displacements is not implemented.

Options for Computing the CPCM terms

In CPCM calculations, there are three major steps. The first step is the calculation of the electrostatic potential created by the quantum object on the surface grid. This step is characterized by:

$$V(\mathbf{r}_g) = \sum_A \frac{Z_A}{|\mathbf{R}_A - \mathbf{r}_g|} - \sum_{\mu\nu} P_{\mu\nu} \int \frac{\mu(\mathbf{r})\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}_g|} d\mathbf{r}$$

$$= \sum_A \frac{Z_A}{|\mathbf{R}_A - \mathbf{r}_g|} - \sum_{\mu\nu} P_{\mu\nu} (\mu\nu|g)$$

The first sum is over atomic nuclei, the second over basis functions. The second step in the calculation consists of converting the surface potential into surface screening charges $\mathbf{q}$. We are not writing it out here, since this step is slightly different for each variant of CPCM. In a nutshell, it involves solving a linear equation system of the dimension of the number of surface segments which is usually done by Cholesky decomposition.

The third, step is to sum the interaction of the surface charges with the quantum electrons into the Fock matrix. This step is given by:

$$F_{\mu\nu}^{(CPCM)} = -\sum_g q_q \int \frac{\mu(\mathbf{r})\,\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}_g|} d\mathbf{r}$$

$$= -\sum_g q_q (\mu\nu|g)$$

Here the sum is over surface segments with surface screening charges q_g.

The time consuming step in these calculations is the calculation of the three-center electrostatic integrals $(\mu\nu|g)$. By default, these integrals are calculated using efficient analytic integration provided by SHARK. However, there are the following alternatives available:

1) The integrals can be evaluated by numeric integration
2) The potential and Fock matrix contributions can be evaluated using the RI density and the RI approximation respectively
3) The potential and Fock matrix elements can be evaluated using multipole approximations to the integrals.

These options are triggered by:

```
%cpcm UseAnalyticOnRebuild true # if true, full Fock matrix builds
                                # are done with analytic integrals
                                # (strongly recommended)

Vopt v_analytic
 # = v_multipole
```

(continues on next page)

(continued from previous page)

```
 # = v_RI
 # = v_numeric
Fopt f_analytic
 # = f_multipole
 # = f_RI
 # = f_numeric
```

Note

- Our experience indicates that analytic integration is still the safest and most accurate options.
- Analytic integration will win out over numerical integration or multipole approximations
- Some savings can be realized with the RI approximation, but it may also introduce non-negligible errors.

2.13.2 Bug Fixes and Extensions

- Relativistic GIAO pVp integrals with finite nucleus models for NMR
- Quadrupole Derivative Integrals for efficient native XTB Gradients
- Prevent errors in general contraction calculations by automatic switching to PGC
- LeanSCF now always calculates a final Fock matrix at the end in order to guarantuee that the energy, orbitals, orbital energies, density and orbitals are all in sync.

2.13.3 The Conductor-like Screening Solvation Model (COSMO)

Note

The COSMO solvation model has been removed from ORCA v4.0.0 !!!

Please use the C-PCM solvation model in combination with the COSMO epsilon function if required!
As a short form to use the C-PCM model with the COSMO epsilon function, you can specify the solvent via `!CPCMC(solvent)`.

2.13.4 The SMD Solvation Model

The SMD solvation model has been proposed by the Cramer and Truhlar groups,[141] and is based on the quantum mechanical charge density of a solute molecule interacting with a continuum description of the solvent. In the model the full solute electron density is used without defining partial atomic charges and the solvent is not represented explicitly but rather as a dielectric medium with the surface tension at the solute–solvent boundary. SMD is a universal solvation model, in the sense that it is applicable to any charged or uncharged solute in any solvent or liquid medium for which a few key descriptors are known. In particular, these descriptors are the dielectric constant, refractive index, bulk surface tension, and acidity and basicity parameters. Neglecting the concentration contribution, the model separates the observable solvation free energy into two main components,

$$\Delta G_{\mathrm{S}} = \Delta G_{\mathrm{ENP}} + \Delta G_{\mathrm{CDS}}. \tag{2.59}$$

In ORCA, the first component is the bulk electrostatic contribution arising from a self-consistent reaction field treatment that involves the electrostatic interaction using the Conductor-like Polarizable Continuum Model (C-PCM). However, the radii are set to "intrinsic atomic Coulomb radii". The second component, called the cavity-dispersion solvent-structure (CDS) term, is the contribution resulting from short-range interactions between the solute and solvent molecules in the first solvation shell. This contribution is a sum of terms that are proportional (with geometry-dependent proportionality constants called atomic surface tensions) to the solvent-accessible surface areas of the

individual atoms of the solute. The CDS contribution to the free energy of solvation is given by

$$\Delta G_{\text{CDS}} = \sum_{k}^{\text{atoms}} \sigma_k A_k(\mathbf{R}, R_{Z_k} + r_s) + \sigma^{[\text{M}]} \sum_{k}^{\text{atoms}} A_k(\mathbf{R}, R_{Z_k} + r_s),$$

where σ_k and $\sigma^{[\text{M}]}$ are the atomic surface tension of atom k and the molecular surface tension, respectively, and A_k is the solvent-accessible surface area (SASA). The SASA depends on the geometry $\mathbf{R}$, the set R_{Z_k} of all atomic van der Waals radii, and the solvent radius r_s, which is added to each of the atomic van der Waals radii. In the program Bondi radii are used for CDS contribution.

More details can be found in the original paper of Marenich et al. [141], which should be cited in publications using results of SMD calculations (together with ref. [140], describing the implementation of the C-PCM in ORCA, which is used by SMD).

SMD can be employed in single point calculations and geometry optimizations, using single-determinant SCF (HF and DFT) and CASSCF methods. In post-SCF methods the result is corrected in the reference wave function. The SMD solvation model is invoked in the input file via

```
! SMD(solvent)
```

where `solvent` is one of the 179 solvents in the SMD library (see Table Table 2.56). Alternatively, one can request the SMD model via the `%cpcm` block by writing:

```
%cpcm
  smd true             # turn on SMD
  SMDsolvent "solvent" # specify the name of solvent from the list
end
```

Independently on the way the user invokes the SMD model, ORCA automatically sets a number of default SMD parameters for the chosen solvent. If required, the user can also manually specify the solvent descriptors used in an SMD calculation in the `%cpcm` block.

```
%cpcm
  soln      # index of refraction at optical frequencies at 293 K
  soln25    # index of refraction at optical frequencies at 298 K
  sola      # Abraham's hydrogen bond acidity
  solb      # Abraham's hydrogen bond basicity
  solg      # relative macroscopic surface tension
  solc      # aromaticity, fraction of non-hydrogenic solvent atoms
            # that are aromatic carbon atoms
  solh      # electronegative halogenicity, fraction of non-hydrogenic
            # solvent atoms that are F, Cl, or Br
 end
```

Let's consider the following input for a water molecule solvated by water,

```
! B3LYP def2-SVP tightscf SMD(water)

* xyz 0 1
O  -0.00000018976103    0.00606010894837    0.00000000004527
H   0.76098169249695   -0.58891312953082   -0.00000000000022
H  -0.76098151333900   -0.58891299029372   -0.00000000000022
*
```

Before the SCF part starts, the program prints the SMD information. This part reads as:

```
--------------------
CPCM SOLVATION MODEL
--------------------
CPCM parameters:
  Epsilon                                         ...      78.3550
  Refrac                                          ...       1.3328
```

(continues on next page)

(continued from previous page)

```
  Rsolv                                          ...       1.3000
  Surface type                                   ... GAUSSIAN VDW
  Discretization scheme                          ... Constant charge density
    Threshold for H atoms                        ...       5.0000 (charges/Ang^2)
    Threshold for non-H atoms                    ...       5.0000 (charges/Ang^2)
  Epsilon function type                          ...         CPCM
Solvent:                                         ... WATER
SMD-CDS solvent descriptors:
  Soln                                           ...       1.3328
  Soln25                                         ...       1.3323
  Sola                                           ...       0.0000
  Solb                                           ...       0.0000
  Solg                                           ...       0.0000
  Solc                                           ...       0.0000
  Solh                                           ...       0.0000
Radii:
  Scheme                                         ... Element-dependent radii
  Radius for O  used is    2.8724 Bohr (=   1.5200 Ang.)
  Radius for H  used is    2.2677 Bohr (=   1.2000 Ang.)
Calculating surface                              ...        done! (  0.0s)
Cavity surface points                            ...          244
Cavity Volume                                    ...     130.0616
Cavity Surface-area                              ...     129.3354
Calculating surface distance matrix              ...        done! (  0.0s)
Performing Cholesky decomposition & store        ...        done! (  0.0s)
Overall time for CPCM initialization             ...                 0.0s
```

After the SCF is converged, the SMD contribution to the total energy is printed (this term is labelled as "SMD CDS" term).

```
               ****************************************************
               *                     SUCCESS                      *
               *           SCF CONVERGED AFTER  10 CYCLES         *
               ****************************************************

Recomputing exchange energy using gridx3       ... done (    0.056 sec)
Old exchange energy                            :     -1.791736873 Eh
New exchange energy                            :     -1.791694663 Eh
Exchange energy change after final integration :      0.000042211 Eh
Total energy after final integration           :    -76.335230273 Eh

SMD CDS free energy correction energy :                 1.44927     Kcal/mol
Total Energy after SMD CDS correction =               -76.332920707 Eh
             **** ENERGY FILE WAS UPDATED (water.en.tmp) ****

----------------
TOTAL SCF ENERGY
----------------

Total Energy       :        -76.33292070695327 Eh           -2077.12437 eV

Components:
Nuclear Repulsion  :          9.11286213981824 Eh             247.97359 eV
Electronic Energy  :        -85.43200537714571 Eh           -2324.72305 eV
One Electron Energy:       -123.06209110061025 Eh           -3348.68974 eV
Two Electron Energy:         37.63008572346453 Eh            1023.96669 eV
CPCM Dielectric    :         -0.01612924659271 Eh              -0.43890 eV
SMD CDS (Gcds)     :          0.00230956608299 Eh               0.06285 eV
```

In ORCA, the user can also request the SMD18 model,[157] which involves refined radii for Bromine and Iodine (the radii adopted for the construction of the cavity used in the C-PCM part). The SMD18 model is shown to accurately describe Halogen-bonding interactions in solution. It is requested via the `SMD18` tag in the simple input:

```
! SMD(solvent) SMD18
```

or via the `%cpcm` block:

```
%cpcm
  SMD18 true  # default = false
end
```

Notes:

- If one is interested in the calculation of free energies of solvation using the SMD model, one just needs to compute ΔG_S according to eq (2.59). However, here one should take into account that the SMD model considers the same concentration (1 mol/L) in both the gaseous and solution phases. Then, if a gas-phase standard state of 1 atm is considered, and we want to compare the calculated ΔG_S with its experimental counterpart, a concentration term equal to 1.89 kcal/mol has to be added to the calculated ΔG_S.

2.13.5 Dynamic Radii Adjustment for Continuum Solvation (DRACO)

The DRACO scheme is an approach that improves the performance of implicit solvation models, in particular the accuracy of the calculated solvation free energies.[158] It is based on a dynamic scaling of the original static radii used to describe the atoms/spheres that define the cavity of the solute.

In this approach, the original radii r_i of an atom i is scaled according to

$$r_{i\text{scale}} = f_{i\text{scale}} r_i \tag{2.60}$$

with the scaling factor $f_{i\text{scale}}$ determined from

$$f_{i\text{scale}} = \text{erf}(a_Z(q_{\text{eff},i} - b_Z)) + 1 \tag{2.61}$$

Here, $q_{\text{eff},i}$ corresponds to the effective partial charge and is defined as

$$q_{\text{eff},i} = q_i + k_Z q_i \text{CN}_i \tag{2.62}$$

where q_i is the atomic partial charge. The parameters a_Z, b_Z and k_Z are element-specific parameters, and CN_i is the fractional coordination number.

For oxygen atoms, the $f_{i\text{scale}}$ is corrected via a term that depends on the Abraham's hydrogen bond acidity (α):

$$f^{\text{O}}_{i\text{scale}} = f_{i\text{scale}} + c_{\text{O}}(0.43 - \alpha) \tag{2.63}$$

with c_{O} a parameter that is different for pure C-PCM or SMD. The correction in eq (2.63) is only applied for solvents with $\alpha < 0.43$.

DRACO is parametrized for C-PCM and SMD for the following solvents: acetonitrile, DMSO, methanol, and water. The element-specific parameters in eqs (2.61),(2.62), and (2.63) are available in https://github.com/grimme-lab/DRACO.

The use of DRACO with C-PCM or SMD is triggered via the following tags in the simple input:

```
! CPCM(solvent) DRACO
```

or

```
! SMD(solvent) DRACO
```

Alternatively, it can also be requested in the `%cpcm` block:

```
%cpcm
  DRACO true
end
```

In ORCA, the default scheme to calculate the partial charges in eq (2.62) is the electronegativity-equilibration (EEQ) charge model (D4 case),[159]. However, one can also request the Charge Extended Hückel (CEH) model.[160] The charge scheme within DRACO is controlled by the following tag in the `%cpcm` block:

```
%cpcm
  draco_charges ceh  # default = eeq
end
```

If CEH charges are requested, ORCA needs to run an XTB calculation to generate them. Here, one should have at least the version 6.7.1 of XTB (older versions will not generate the CEH charges, see section *Program Components*). Regarding the fractional coordination number in eq (2.62), it is calculated as described in GFN2-XTB calculations.[161]

DRACO can be used in single-point energy calculations (with both CEH and EEQ charges) and in geometry optimizations (with the EEQ charges)[162]. The DRACO contributions to the analytical Hessian are not yet implemented. In this case, the user should request `numfreq`.

Notice that calculations where DRACO is requested involve a discretization scheme of the surface of the cavity based on a constant number of charges per sphere. The default is 194 Lebedev points. This contrast with the scheme used for C-PCM/SMD calculations without DRACO, where the default is the isodensity scheme, as discussed in Section *Use of the Gaussian Charge Scheme*. This choice is needed, specially for geometry optimizations. For DRACO, the radii of the spheres in the cavity changes in each optimization step. Then, if ORCA used by default the isodensity scheme for DRACO, the associated Lebedev grid per atom could change between two consecutive steps, leading to discontinuities in the total energy and its gradient. Nevertheless, the isodensity scheme can still be requested for calculations involving DRACO via the tag `DRACOisodens` in the `%cpcm` block:

```
%cpcm
  dracoisodens true  # default = false
end
```

2.13.6 OpenCOSMO-RS

ORCA is interfaced to openCOSMO-RS,[142, 163] an open source implementation of the COSMO-RS model.[164, 165] This model is widely used in both academia and industry to predict fluid phase thermodynamics.

The main idea behind COSMO-RS is that the interaction between molecules in the liquid phase can de depicted as an ensemble of interacting surface segments. The properties on the surface segments are calculated via QM calculations of the solute and the solvent in a perfect conductor ($\epsilon = \infty$), and the interaction free energies between segments are functions of a set of descriptors. Among them, the most relevant one is the screening charge density (σ).

A complete description of the COSMO-RS model used in ORCA is provided in refs [142, 163], and the code is available in https://github.com/TUHH-TVT. In particular, the parametrization for ORCA 6.0 is called openCOSMO-RS 24a. The corresponding the executable is shipped together with the ORCA 6.0 binaries and called `openCOSMORS`. ORCA will then call internally this executable whenever it is needed.

COSMO-RS calculations within ORCA are a special type of calculation. By requesting COSMO-RS, ORCA runs a set of single-point-energy calculations for both the solute and the solvent and then calls the openCOSMO-RS executable, which expects an input geometry for the solute and the solvent and calculates the free energy of solvation of the solute in the solvent. A workflow of a calculation requesting COSMO-RS in ORCA is the following:

1. Single-point-energy calculation of the solute in the gas-phase
2. Single-point-energy calculation of the solute in a conductor ($\epsilon = \infty$)
3. Single-point-energy calculation of the solvent in a conductor ($\epsilon = \infty$)
4. Do COSMO-RS via the openCOSMO-RS executable and compute solvation properties

Points 1 to 3 correspond to DFT calculations at the BP86/def2-TZVPD level. This is the level of theory used to parametrize COSMO-RS for ORCA 6.0. As it is shown later, the user can change the functional and basis set, but this is not recommended!

Regarding calculations 2 and 3, to produce smooth σ profiles, ORCA discards the surface segments with an area < 0.01 $Å^2$. For these two calculations, the radii of several elements used to construct the cavity of solute and solvent are different than the defaults employed within non-COSMO-RS calculations (using C-PCM). This is due to the fact that the provided openCOSMO-RS binaries involve a special parametrization of COSMO-RS for ORCA 6.0, and this does not only affect the COSMO-RS parameters, but also the radii of several elements used in the C-PCM part.

After point 4 is done, the free energy of solvation (ΔG_S) of the solute in the solvent is printed in the ORCA output file. In ORCA 6.0 the use of openCOSMO-RS is restricted to the calculation of ΔG_S.

How to Run a ORCA/COSMO-RS Calculation

ORCA/COSMO-RS calculations are controlled through the `%cosmors` block. These type of calculations require two input structures, one for the solute and one for the solvent. The solute coordinates are provided in the input file as done for any other type of calculation. However, regarding the structure for the solvent, there are two options:

1. Retrieve the structure of the solvent from a database
2. Provide the structure in a separate file

To use strategy 1, we need to request the solvent via:

```
COSMORS(Solvent)
```

in the simple input or using the `%cosmors` block:

```
%cosmors
  solvent "Solvent"
end
```

The list of internal solvents available in ORCA are shown in Table 2.56. For instance, for a water molecule solvated by acetonitrile, the input file looks like:

```
!COSMORS(Acetonitrile)
* xyz 0 1
O    0.00000006589375       0.00157184228646       0.00000000004493
H    0.77316868532439      -0.58666889665624      -0.00000000000005
H   -0.77316876182122      -0.58666895650640      -0.00000000000005
*
```

If the user wants to provide a structure for the solvent (strategy 2), then a separate file with extension **.cosmorsxyz** should be available. The name of this file (without extension) is controlled by the tag `solventfilename` in the `%cosmors` block. For instance, if we want to calculate the free energy of solvation of acetone in water, the ORCA input file would look like this:

```
%cosmors
  solventfilename "water"
end

* xyz 0 1
H    1.99757808828569      0.25022586507917      0.72957579847856
C    1.44666788654273      0.06088176074125     -0.18975506731892
H    1.67115809398389      0.82690156694531     -0.93346420198265
H    1.76410010378270     -0.89580894800398     -0.61702931492419
C   -0.03218812888253     -0.00668250402989      0.08117960952503
O   -0.47126229461002     -0.06383203315654      1.21590571064657
C   -0.93671505604705     -0.00759565576779     -1.12174123739234
H   -0.88761294036774      0.97611724740670     -1.59852429171257
H   -0.58940208311204     -0.73299176093053     -1.85998395179737
H   -1.96332366957567     -0.22151553828368     -0.83026305352213
*
```

The structure of the solvent (water) is the one in the `water.cosmorsxyz` file:

```
3
0 1
O    0.00000006589375       0.00157184228646       0.00000000004493
H    0.77316868532439      -0.58666889665624      -0.00000000000005
H   -0.77316876182122      -0.58666895650640      -0.00000000000005
```

where the first line corresponds to the number of atoms, and in the second line the charge and multiplicity are provided.

The output for COSMO-RS is printed in the ORCA output file after the line that reads `OPENCOSMO-RS CALCULATION`. First of all, the information regarding the level of theory, the solute and the solvent is printed. For the example above (acetone in water), it reads as,

```
-----------------------------------------------------------------------------
                           OPENCOSMO-RS CALCULATION
-----------------------------------------------------------------------------

----------------------
GENERAL INFORMATION
----------------------
Calculation method                          ... DFT
Functional                                  ... BP86
Basis set                                   ... DEF2-TZVPD

----------------------
SOLUTE INFORMATION
----------------------
Number of atoms                             ...     10
Total charge                                ...      0
Multiplicity                                ...      1

CARTESIAN COORDINATES (ANGSTROEM)
  H      1.997578    0.250226    0.729576
  C      1.446668    0.060882   -0.189755
  H      1.671158    0.826902   -0.933464
  H      1.764100   -0.895809   -0.617029
  C     -0.032188   -0.006683    0.081180
  O     -0.471262   -0.063832    1.215906
  C     -0.936715   -0.007596   -1.121741
  H     -0.887613    0.976117   -1.598524
  H     -0.589402   -0.732992   -1.859984
  H     -1.963324   -0.221516   -0.830263

----------------------
SOLVENT INFORMATION
----------------------
Solvent name                                ... water
Number of atoms                             ...      3
Total charge                                ...      0
Multiplicity                                ...      1

CARTESIAN COORDINATES (ANGSTROEM)
  O      0.000000    0.001572    0.000000
  H      0.773169   -0.586669   -0.000000
  H     -0.773169   -0.586669   -0.000000
```

After these lines, ORCA prints the final single point energy for each of the QM calculations, together with the output file to which the ORCA output is redirected:

```
---------------------------------------------
Single Point Calculation (solute / gas-phase)
---------------------------------------------
Output single point calculation redirected to >test.solute_vac.lastout
```

(continues on next page)

(continued from previous page)

```
FINAL SINGLE POINT ENERGY (Solute-gas-phase)     -193.233975714789

---------------------------------------------
Single Point Calculation (solute / CPCM)
---------------------------------------------
Output single point calculation redirected to >test.solute_cpcm.lastout

FINAL SINGLE POINT ENERGY (Solute-CPCM)     -193.243987123912

---------------------------------------------
Single Point Calculation (solvent / CPCM)
---------------------------------------------
Output single point calculation redirected to >test.solvent_cpcm.lastout

FINAL SINGLE POINT ENERGY (Solvent-CPCM)      -76.479155022866
```

Once this information is printed, ORCA calls the openCOSMO-RS executable and ΔG_S is printed in the following block:

```
----------------------
SOLVATION DATA
----------------------
Reference temperature              :                 298.15 K
Free energy of solvation (dGsolv)  :       -0.006626234385 Eh             -4.
↪158026 kcal/mol

Note: In order to calculate the free energy of the solvated solute (Gsolv), one␣
↪should add
      the computed dGsolv to the "Final Gibbs free energy" (Gvac = H-T*S) of the␣
↪solute in gas-phase.
      That is: Gsolv = Gvac + dGsolv. Here, the Gvac has been calculated␣
↪previously after a frequency
      calculation of the solute at a certain level of theory and printed in the
↪"THERMOCHEMISTRY" block.
```

As pointed out in the last paragraph of the COSMO-RS output, to calculate the Gibbs free energy of the solute solvated by the given solvent, one should add the calculated ΔG_S to the `Final Gibbs free energy` of the solute in the gas-phase.

Note: The workflow explained above for ORCA/openCOSMO-RS calculations involves the same structure for the solute in the gas-phase and in solution. However, these structures may differ substantially depending on the type of solute and solvent. In ORCA, it is possible to optimize the structures for each of the three calculations needed in the ORCA/openCOSMO-RS workflow. That is, (1) the solute in gas-phase, (2) the solute in a conductor, and (3) the solvent in a conductor. To do that, one needs to add the level of optimization via the `dftfunc` tag, which is exclusive for the DFT functional, but as an exception, can be extended with the optimization tag:

```
%cosmors
  dftfunc  "BP86 tightopt"
end
```

OpenCOSMO-RS Keywords

The parameters that can be defined in the `%cosmors` block in the ORCA input file are the following:

```
%cosmors
  aeff                 5.92500 # Effective contact area between surface segments␣
↪(Å^2)
  lnalpha              0.20200 # Logarithm of the misfit prefactor
  lnchb                0.16600 # Hydrogen bond (HB) strength parameter
  chbt                 1.50    # Parameter for the temperature dependence of the HB
  sigmahb              9.61e-3 # HB threshold parameter (e/Å^2)
  rav                  0.50    # Radius to average ideal screening charges in Å
  fcorr                2.40    # Parameter adjusted from dielectric screening␣
↪energies
  ravcorr              1.00    # Additional radius to calculate the misfit energy␣
↪in Å
  astd                 41.6240 # Standard surface area (normalization factor) in Å^
↪2
  zcoord               10.0    # Coordination number
  dgsolv_eta          -4.44800 # Offset for the solv. energy calculation
  dgsolv_omegaring     0.26300 # Correction for solv. energy of molecules with␣
↪rings
  temp                 298.15  # Reference temperature in Kelvin
  dftfunc              "BP86"  # String for the DFT functional
  dftbas               "def2-TZVPD"  # String for the basis set
  solvent              "THF"   # Solvent from the internal database
  solventfilename      "water" # Name of the .cosmorsxyz solvent file
  orbs_vac             false   # Reuse the gas-phase orbitals for the calculation
                               # involving the solute in a conductor
end
```

It is not recommended to change the defaults of the COSMO-RS parameters.

2.13.7 Implicit Solvation in Coupled-Cluster Methods

The coupled-cluster Lagrangian, $\mathcal{L}$, for a system implicitly solvated reads as follows,[166, 167, 168]

$$\mathcal{L}(\Lambda, T) = \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}H_0\mathrm{e}^{T}|\psi_0\rangle + \frac{1}{2}\mathbf{Q}(\Lambda, T)\cdot\mathbf{V}(\Lambda, T) \tag{2.64}$$

where ψ_0 is the reference wave function, and H_0 is the Hamiltonian for the isolated molecule. The operator T for CCSD is defined in terms of single and double excitations ($T = T_1 + T_2$), and Λ is the de-excitation operator, defined in terms of the Lagrange multipliers:

$$T = T_1 + T_2 = \sum_{ia} t_a^i a_a^+ a_i + \sum_{ijab} t_{ab}^{ij} a_a^+ a_b^+ a_j a_i \tag{2.65}$$

$$\Lambda = \sum_{ia} \lambda_i^a a_i^+ a_a + \frac{1}{2}\sum_{ijab} \lambda_{ij}^{ab} a_i^+ a_a a_j^+ a_b \tag{2.66}$$

Here, t_a^i and t_{ab}^{ij} are the singles and doubles wave function amplitudes and a_i and a_a^+ are standard fermion annihilation and creation operators, respectively. Canonical occupied orbitals are denoted by the symbols $i, j, k, \ldots$, virtual orbitals by the symbols $a, b, c, \ldots$, and we use the symbols $p, q, r \ldots$ for general orbital indices.

The quantities $\mathbf{Q}$ and $\mathbf{V}$ are the CC expectation values of the C-PCM operators $\mathbf{Q}$ and $\mathbf{V}$, which are the solvation charges vector and solute potential vector defined at the position the charges, respectively.

$$\mathbf{Q}(\Lambda, T) = \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}\mathbf{Q}\mathrm{e}^{T}|\psi_0\rangle \tag{2.67}$$

$$\mathbf{V}(\Lambda, T) = \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}\mathbf{V}\mathrm{e}^{T}|\psi_0\rangle \tag{2.68}$$

Equation (2.64) can be rewritten by introducing the normal product form of an operator:

$$X_N = X - \langle\psi_0|X|\psi_0\rangle = X - X_0 \tag{2.69}$$

If one uses this result in eq (2.64), together with the fact that $\mathbf{Q}$ and $\mathbf{V}$ are related through eq (2.54), then eq (2.64) reads as,

$$\begin{aligned}\mathcal{L}(\Lambda,T) =& \langle\psi_0|H_0|\psi_0\rangle + \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}H_{0N}\mathrm{e}^{T}|\psi_0\rangle + \frac{1}{2}\mathbf{Q}_0\cdot\mathbf{V}_0 + \mathbf{Q}_0\cdot\mathbf{V}_N(\Lambda,T) + \frac{1}{2}\mathbf{Q}_N(\Lambda,T)\cdot\mathbf{V}_N(\Lambda,T) = \\ =& E_0 + \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}H_{0N}\mathrm{e}^{T}|\psi_0\rangle + \mathbf{Q}_0\cdot\mathbf{V}_N(\Lambda,T) + \frac{1}{2}\mathbf{Q}_N(\Lambda,T)\cdot\mathbf{V}_N(\Lambda,T)\end{aligned} \tag{2.70}$$

Here, $\mathbf{Q}_0$ and $\mathbf{V}_0$ are the $\mathbf{Q}$ and $\mathbf{V}$ vectors calculated with the ψ_0 wave function, and E_0 is the reference energy ($E_0 = \langle\psi_0|H_0|\psi_0\rangle + \frac{1}{2}\mathbf{Q}_0\cdot\mathbf{V}_0$). Different approximations can be adopted in eq (2.70) depending on how one calculates its last term $\frac{1}{2}\mathbf{Q}_N(\Lambda,T)\cdot\mathbf{V}_N(\Lambda,T)$.

In ORCA there are three different CCSD/CPCM approaches: (i) the PTE scheme, (ii) the PTE(S) scheme, and the (iii) the PTES scheme, being the last one the default. Here, the acronym PTE stands for "perturbation theory and energy" and "S" for singles. The choice of any of these approaches is controlled via the tag `CPCMccm`. Information about which CCSD/C-PCM is used by ORCA in a calculation is printed in the `ORCA-MATRIX DRIVEN CI` block in the output file in the line starting by `CPCM scheme`:

```
-------------------------------------------------------------------------------
                             ORCA-MATRIX DRIVEN CI
-------------------------------------------------------------------------------

--------------------------------
AUTOMATIC CHOICE OF INCORE LEVEL
--------------------------------

Memory available                           ...   2000.00 MB
Memory needed for S+T                      ...      9.26 MB
Memory needed for J+K                      ...     18.57 MB
Memory needed for DIIS                     ...    129.61 MB
Memory needed for 3-ext                    ...     72.69 MB
Memory needed for 4-ext                    ...    486.14 MB
Memory needed for triples                  ...      0.00 MB
 -> Final InCoreLevel    ... 5

Wavefunction type
-----------------
Correlation treatment                      ...      CCSD
Single excitations                         ... ON
Orbital optimization                       ... OFF
Calculation of Lambda equations            ... ON
Calculation of Brueckner orbitals          ... OFF
Perturbative triple excitations            ... OFF
CPCM scheme                                ... PTE(S)
Calculation of F12 correction              ... OFF
```

In the following subsections, we describe the different CCSD/C-PCM approaches available in ORCA and how to use them.

PTE scheme

In the "perturbation theory energy" (PTE) scheme, the last term in eq (2.70) is equal to zero (this term does not depend on Λ and T),

$$\mathcal{L}(\Lambda, T) = E_0 + \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}H_{0N}\mathrm{e}^{T}|\psi_0\rangle + \mathbf{Q}_0 \cdot \mathbf{V}_N(\Lambda, T) \tag{2.71}$$

The potential $\mathbf{V}_N$ can be written as follows:

$$\mathbf{V}_N(\Lambda, T) = \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}\sum_{pq}\mathbf{v}_{pq}\{p^+q\}\mathrm{e}^{T}|\psi_0\rangle = \sum_{pq}\mathbf{v}_{pq}\langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}\{p^+q\}\mathrm{e}^{T}|\psi_0\rangle = \sum_{pq}\mathbf{v}_{pq}\Gamma_{pq} \tag{2.72}$$

where we have used that $\mathbf{V}_N = \sum_{pq}\mathbf{v}_{pq}\{p^+q\}$ (second-quantized form of a normal ordered operator), with $\mathbf{v}_{pq}$ the components of the solute potential in the MO basis. The matrix Γ is the CCSD relaxed one-electron density matrix. Then, the contribution to the equations for the T amplitudes comes from the derivative of $\mathbf{V}_N(\Lambda, T)$ with respect to the Λ amplitudes ($\mathbf{Q}_0$ does not depend on the Lagrange multipliers). In this context, the Hamiltonian H_{0N} contains a term that depends on the elements of the Fock matrix ($\sum_{pq} f_{pq}\{p^+q\}$) and that has the same functional form as $\mathbf{V}_N$. As the Fock matrix is updated in the reference calculation with a C-PCM term that reads as (in AO basis) $F_{\mu\nu}^{\mathrm{CPCM}} = \mathbf{Q}_0 \cdot \mathbf{V}_{\mu\nu}$, then the term $\mathbf{Q}_0 \cdot \mathbf{V}_N(\Lambda, T)$ is added implicitly to eq (2.71).

Once the T amplitudes are obtained, the total energy, E, is calculated as

$$E = E_0 + \langle\psi_0|\mathrm{e}^{-T}(H_{0N} + \mathbf{Q}_0 \cdot \mathbf{V}_N)\mathrm{e}^{T}|\psi_0\rangle = E_0 + \sum_{ia}F_{ia}t_a^i + \frac{1}{4}\sum_{ijab}\langle ij||ab\rangle\left(t_{ab}^{ij} + 2t_a^i t_b^j\right) \tag{2.73}$$

Then, the C-PCM contribution to the CC energy within the PTE scheme occurs through the term $\frac{1}{2}\mathbf{Q}_0 \cdot \mathbf{V}_0$ in E_0 and implicitly through the Fock matrix elements F_{ia} ($F_{ia} = F_{ia}^0 + \mathbf{Q}_0 \cdot \mathbf{v}_{ia}$).

The PTE scheme corresponds to `CPCMccm 0`, and is implemented for canonical-CCSD (RHF and UHF) and DLPNO-CCSD (RHF and UHF). For instance, for a DLPNO-CCSD calculation (closed-shell reference) of a system solvated by water using the PTE scheme, the input file looks like:

```
! DLPNO-CCSD cc-pVTZ cc-PVTZ/C TightSCF CPCM(Water)

%cpcm
  CPCMccm 0
end

* xyzfile 0 1 water.xyz
```

PTE(S) scheme

In this scheme (where the "S" stands for singles), the last term in eq (2.70) depends on the T amplitudes, but not on the Λ amplitudes,

$$\mathcal{L}(\Lambda, T) = E_0 + \langle\psi_0|(1+\Lambda)\mathrm{e}^{-T}H_{0N}\mathrm{e}^{T}|\psi_0\rangle + \mathbf{Q}_0 \cdot \mathbf{V}_N(\Lambda, T) + \frac{1}{2}\mathbf{Q}_N(T) \cdot \mathbf{V}_N(T) \tag{2.74}$$

Again, in the same way as for the PTE scheme, the C-PCM contribution to the equations for the T amplitudes comes from the term $\mathbf{Q}_0 \cdot \mathbf{V}_N(\Lambda, T)$ in eq (2.74), which is implictly added to the Fock matrix elements in the MO basis. The last term in eq (2.74) does not depend on the Λ amplitudes and then does not contribute to the equations for the T amplitudes. However, this term depends on the T amplitudes through the elements γ_{ai} of the density matrix Γ,

$$\gamma_{ai} = t_a^i + \sum_{me}\left(t_{ae}^{im} - t_e^i t_a^m\right)\lambda_m^e - \frac{1}{2}\sum_{mnef}\lambda_{mn}^{ef}\left(t_{ef}^{in}t_a^m + t_{af}^{mn}t_e^i\right) \tag{2.75}$$

and then it contributes to the final energy E in the following way:

$$\frac{1}{2}\mathbf{Q}_N(T) \cdot \mathbf{V}_N(T) = \frac{1}{2}\left(\sum_{ai}t_a^i\mathbf{q}_{ai}\right) \cdot \left(\sum_{ai}t_a^i\mathbf{v}_{ai}\right) \tag{2.76}$$

That gives the final equation for the total energy of our system,

$$
\begin{aligned}
E &= E_0 + \langle\psi_0|\mathrm{e}^{-T}(H_{0N} + \mathbf{Q}_0 \cdot \mathbf{V}_N)\mathrm{e}^{T}|\psi_0\rangle + \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left(\sum_{ai} t_a^i \mathbf{v}_{ai}\right) = \\
&= E_0 + \sum_{ia} F_{ia} t_a^i + \frac{1}{4}\sum_{ijab}\langle ij||ab\rangle \left(t_{ab}^{ij} + 2t_a^i t_b^j\right) + \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left(\sum_{ai} t_a^i \mathbf{v}_{ai}\right)
\end{aligned}
\tag{2.77}
$$

Therefore, the CC energy for a solvated system within the PTE(S) scheme involves three C-PCM contributions: (1) the term $\frac{1}{2}\mathbf{Q}_0 \cdot \mathbf{V}_0$ included in E_0, (2) the term $\sum_{ia} \mathbf{Q}_0 \cdot \mathbf{v}_{ia} t_a^i$ that occurs implicitly through $\sum_{ia} F_{ia} t_a^i$ and (3) the term $\frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left(\sum_{ai} t_a^i \mathbf{v}_{ai}\right)$.

This scheme is available in ORCA for canonical-CCSD (RHF and UHF) and DLPNO-CCSD (RHF and UHF). In order to use it, the user needs to add the tag `CPCMccm 1` in the `%cpcm` block.

```
! DLPNO-CCSD cc-pVTZ cc-PVTZ/C TightSCF CPCM(Water)

%cpcm
  CPCMccm 1
end

* xyzfile 0 1 water.xyz
```

PTES scheme

In this scheme, both $\mathbf{Q}_N$ and $\mathbf{V}_N$ in the last term in eq (2.70) depend on the T amplitudes but just one of them depends on the Λ amplitudes,

$$
\mathcal{L}(\Lambda, T) = E_0 + \langle\psi_0|(1+\Lambda)\mathrm{e}^{T} H_{0N} \mathrm{e}^{T}|\psi_0\rangle + \mathbf{Q}_0 \cdot \mathbf{V}_N(\Lambda, T) + \frac{1}{2}\mathbf{Q}_N(T) \cdot \mathbf{V}_N(\Lambda, T) \tag{2.78}
$$

The C-PCM terms enter these equations on the one hand, implicitly, through the elements of the Fock matrix (like for the PTE and PTE(S) schemes), and on the other hand, explicitly through the derivatives of $\frac{1}{2}\mathbf{Q}_N(T) \cdot \mathbf{V}_N(\Lambda, T)$ with respect to Λ. If we call $\mathcal{L}_{\text{CPCM}}$ the last C-PCM term in eq (2.78), then the contribution from this term to the T amplitudes equations read as:

$$
\frac{\partial \mathcal{L}_{\text{CPCM}}}{\partial \lambda_i^a} = \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left[-\sum_{j} t_a^j \mathbf{v}_{ji} + \sum_{b} t_b^i \mathbf{v}_{ab} + \mathbf{v}_{ia} + \sum_{bj}\left(t_{ba}^{ji} - t_a^j t_b^i\right)\mathbf{v}_{bj}\right] \tag{2.79}
$$

$$
\frac{\partial \mathcal{L}_{\text{CPCM}}}{\partial \lambda_{ij}^{ab}} = \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left[-\frac{1}{2}\sum_{k} t_{ab}^{kj} \mathbf{v}_{ki} + \frac{1}{2}\sum_{c} t_{ac}^{ij} \mathbf{v}_{bc} - \frac{1}{2}\sum_{ck}\left(t_{ab}^{kj} t_c^i + t_{cb}^{ij} t_a^k\right)\mathbf{v}_{ck}\right] \tag{2.80}
$$

The contribution to the energy is the same as that for the PTE(S) scheme, but with different values for the T amplitudes (as the equations to calculate them differ slightly from those for the PTE(S) scheme).

$$
\begin{aligned}
E &= E_0 + \langle\psi_0|\mathrm{e}^{-T}(H_{0N} + \mathbf{Q}_0 \cdot \mathbf{V}_N)\mathrm{e}^{T}|\psi_0\rangle + \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left(\sum_{ai} t_a^i \mathbf{v}_{ai}\right) = \\
&= E_0 + \sum_{ia} F_{ia} t_a^i + \frac{1}{4}\sum_{ijab}\langle ij||ab\rangle \left(t_{ab}^{ij} + 2t_a^i t_b^j\right) + \frac{1}{2}\left(\sum_{ai} t_a^i \mathbf{q}_{ai}\right) \cdot \left(\sum_{ai} t_a^i \mathbf{v}_{ai}\right)
\end{aligned}
\tag{2.81}
$$

This scheme is the default CCSD/C-PCM approach in ORCA and is available in ORCA for canonical-CCSD (RHF and UHF) and DLPNO-CCSD (RHF and UHF). In this case, the tag `CPCMccm` in the `%cpcm` block is equal to 2. However, as the PTES scheme is the default in ORCA, the user just needs to add the information about the solvent in the input file, in order to use this approach.

```
! DLPNO-CCSD cc-pVTZ cc-PVTZ/C TightSCF CPCM(Water)

* xyz 0 1
O  -0.00000018976103    0.00606010894837    0.00000000004527
H   0.76098169249695   -0.58891312953082   -0.00000000000022
H  -0.76098151333900   -0.58891299029372   -0.00000000000022
*
```

Notes regarding the use of the CCSD/C-PCM schemes:

- For calculations within the PTE(S) and the PTES schemes, the explicit C-PCM contribution to the total energy (see eqs. (2.77) and (2.81)) is printed in the `COUPLED CLUSTER ENERGY` block after the equations for the "T" amplitudes converge. In this case, this energy is labelled as `C-PCM corr. term` and is already included in the correlation energy, `E(CORR)`. For the input example from above, this information looks like:

```
----------------------
COUPLED CLUSTER ENERGY
----------------------

E(0)                                          ...    -76.066903687
E(CORR)(strong-pairs)                         ...     -0.267203798
E(CORR)(weak-pairs)                           ...     -0.000106106
E(CORR)(corrected)                            ...     -0.267309904
C-PCM corr. term (included in E(CORR))        ...     -0.000003158
E(TOT)                                        ...    -76.334213591
Singles Norm <S|S>**1/2                       ...      0.017784967
T1 diagnostic                                 ...      0.006287935
```

  This contribution does not represent the whole C-PCM contribution to the correlation energy, as this one also occurs, implicitly, through the "T" amplitudes.

- The C-PCM contribution to the Λ equations is implemented in ORCA for the PTE(S) and PTES schemes. Then, the user can request unrelaxed densities.

2.13.8 Complete Keyword List for the `%cpcm` Block

The available parameters/options that the user can request via the `%cpcm` block are:

```
%cpcm
  epsilon             80.0    # Dielectric constant
  refrac               1.0    # Refractive index
  rsolv                1.3    # Solvent probe radius in Angstroem (for SES-type␣
↪cavities)
  rmin                 0.5    # Minimal GEPOL sphere radius in Angstroem
  pmin                 0.1    # Minimal distance between two surface points in␣
↪Angstroem
  fepstype            cpcm    # Epsilon function type: cpcm, cosmo
  xfeps                0.0    # X parameter for the feps scaling function
  surfacetype vdw_gaussian    # Cavity surface: gepol_ses, gepol_sas
                                                vdw_gaussian, gepol_ses_gaussian
  ndiv                   5    # Maximum depth for recursive sphere generation
  num_leb              302    # Lebedev points for the Gaussian charge scheme
  radius[N]            1.3    # Atomic radius for atomic number N in Angstroem
  AtomRadii(N,1.4)            # Atomic radius for the Nth atom in Angstroem
  scale_gauss          1.2    # Scaling factor for the atomic radii in the
                                Gaussian charge scheme
  cut_area             0.0    # Cutoff for the area of a surface segment in a.u.
                                Only valid for the Gaussian charge scheme
  cut_swf             1e-7    # Cutoff for the switching function
                                Only valid for the Gaussian charge scheme
  thresh_h             5.0    # Threshold for the charge density on a hydrogen
```

(continues on next page)

(continued from previous page)

```
                                  atom in charges/Å^2 (isodensity scheme)
  thresh_noth          5.0     # Threshold for the charge density on non-hydrogen
                                  atoms in charges/Å^2 (isodensity scheme)
  solvent          "water"     # Solvent name (for C-PCM)
  smd                 true     # Turn on SMD
  smdsolvent      "ethanol"    # Solvent name for SMD
  soln               1.3611    # Index of refraction at optical frequencies at 293 K
  soln25             1.3593    # Index of refraction at optical frequencies at 298 K
  sola                 0.37    # Abraham's hydrogen bond acidity
  solb                 0.48    # Abraham's hydrogen bond basicity
  solg                31.62    # Relative macroscopic surface tension
  solc                 0.00    # Aromaticity, fraction of non-hydrogenic solvent␣
↪atoms
                               # that are aromatic carbon atoms
  solh                 0.00    # Electronegative halogenicity, fraction of non-
↪hydrogenic
                               # solvent atoms that are F, Cl, or Br
  SMD18               false    # Turn on SMD18
  CPCMccm                 0    # Coupled-cluster/C-PCM scheme (default = 2)
  cds_cpcm                0    # Use of the GVDW_nel or GSES_nel scheme (default =␣
↪0)
  draco                true    # Turn on DRACO (default = false)
  dracoisodens        false    # Use isodensity scheme for DRACO (default = false)
  draco_charges         ceh    # Charges used within DRACO (default = eeq)
  vopt           v_analytic    # Method to calculate the ESP at the position of the␣
↪charges:
                               #    v_analytic (default): via the analytic ESP␣
↪integrals
                               #    v_numeric: via numerical integration
                               #    v_ri: via the RI-density
                               #    v_multipole: via BUPO
 fopt            f_analytic    # Method to calculate the C-PCM contribution to the␣
↪Fock
                               # or Kohn-Sham matrix:
                               #    f_analytic (default): via the ESP integrals
                               #    f_numeric: via numerical integration
                               #    f_ri: via the RI-density
                               #    f_multipole: via BUPO
end
```

2.14 Integral Handling

As the number of nonzero integrals grows very rapidly and reaches easily hundreds of millions even with medium sized basis sets in medium sized molecules, storage of all integrals is not generally feasible. This desperate situation prevented SCF calculations on larger molecules for quite some time, so that Almlöf [169, 170, 171] made the insightful suggestion to repeat the integral calculation, which was already the dominant step, in *every SCF cycle* to solve the storage problem. Naively, one would think that this raises the effort for the calculation to $n_{\text{iter}} t_{\text{integrals}}$ (where n_{iter} is the number of iterations and $t_{\text{integrals}}$ is the time needed to generate the nonzero integrals). However, this is not the case because only the change in the Fock matrix is required from one iteration to the next, but not the Fock matrix itself. As the calculations starts to converge, more and more integrals can be skipped. The integral calculation time will still dominate the calculation quite strongly, so that ways to reduce this burden are clearly called for. As integrals are calculated in *batches*[1] the cost of evaluating the given batch of shells p, q, r, s may be estimated as:

$$\text{cost} \approx n_p n_q n_r n_s \left(2l_p + 1\right)\left(2l_q + 1\right)\left(2l_r + 1\right)\left(2l_s + 1\right) \tag{2.82}$$

[1] A *batch* is a set of integrals that arises from all components of the shells involved in the integral. For example a $\langle pp|pp\rangle$ batch gives rise to $3 \times 3 \times 3 \times 3 = 81$ integrals due to all possible combinations of p_x, p_y and p_z functions in the four shells. Computations based on batches lead to great computational advantages because the 81 integrals involved in the $\langle pp|pp\rangle$ batch share many common intermediate quantities.

Here, n_p is the number of primitives involved in shell p, and l_p is the angular momentum for this shell. *Large* integrals are also good candidates for storage, because small changes in the density that multiply large integrals are likely to give a nonzero contribution to the changes in the Fock matrix.

ORCA thus features two possibilities for integral handling, which are controlled by the variable `SCFMode`. In the mode `Conventional`, all integrals above a given threshold are stored on disk (in a compressed format that saves much disk space). In the mode `Direct`, all two-electron integrals are recomputed in each iteration.

Two further variables are of importance: In the `Conventional` mode the program may write enormous amounts of data to disk. To ensure this stays within bounds, the program first performs a so-called "statistics run" that gives a pessimistic estimate of how large the integral files will be. Oftentimes, the program will overestimate the amount of disk space required by a factor of two or more. The maximum amount of disk space that is allowed for the integral files is given by `MaxDisk` (in Megabytes).

On the other hand, if the integral files in `Conventional` run are small enough to fit into the central memory, it is faster to do this since it avoids I/O bottlenecks. The maximum amount of memory allocated for integrals in this way is specified by `MaxIntMem` (in Megabytes). If the integral files are larger than `MaxIntMem`, no integrals will be read into memory.

```
%scf
  MaxIter   100    # Max. no. of SCF iterations
  SCFMode Direct   # default, other choice: Conventional
  Thresh  1e-8     # Threshold for neglecting integrals / Fock matrix contributions
                   # Depends on the chosen convergence tolerance (in Eh).
  TCut    1e-10    # Threshold for neglecting primitive batches. If the prefactor
                   # in the integral is smaller than TCut, the contribution of the
                   # primitive batch to the total batch is neglected.
  DirectResetFreq 20   # default: 15
  MaxDisk  2500        # Max. amount of disk for 2 el. ints. (MB)
  MaxIntMem 400        # Max. amount of RAM for 2 el. ints. (MB)
end
```

The value of `DirectResetFreq` sets the number of incremental Fock matrix builds after which the program should perform a full Fock matrix build in a `Direct` SCF calculation. To prevent numerical instabilities that arise from accumulated errors in the recursively build Fock matrix, the value should not be too large, since this will adversely affect the SCF convergence. If the value is too small, the program will update more frequently, but the calculation will take considerably longer, since a full Fock matrix build is more expensive than a recursive one.

The thresholds `TCut` and `Thresh` also deserve a closer explanation. `Thresh` is a threshold that determines when to neglect two-electron integrals. If a given integral is smaller than `Thresh` Eh, it will not be stored or used in Fock matrix construction. Additionally, contributions to the Fock matrix that are smaller than `Thresh` Eh will not be calculated in a `Direct` SCF.

Clearly, it would be wasteful to calculate an integral, then find out it is good for nothing and thus discard it. A useful feature would be an efficient way to estimate the size of the integral *before it is even calculated*, or even have an estimate that is a *rigorous upper bound* on the value of the integral. Häser and Ahlrichs [172] were the first to recognize that such an upper bound is actually rather easy to calculate. They showed that:

$$|\langle ij\,|kl\rangle| \leqslant \sqrt{\langle ij\,|ij\rangle}\sqrt{\langle kl\,|kl\rangle} \tag{2.83}$$

where:

$$\langle ij\,|kl\rangle = \int\int \phi_i\left(\vec{r}_1\right)\phi_j\left(\vec{r}_1\right) r_{12}^{-1}\phi_k\left(\vec{r}_2\right)\phi_l\left(\vec{r}_2\right) d\vec{r}_1 d\vec{r}_2 \tag{2.84}$$

Thus, in order to compute an upper bound for the integral only the right hand side of this equation must be known. This involves only two index quantities, namely the matrix of two center exchange integrals $\langle ij\,|ij\rangle$. These integrals are easy and quick to calculate and they are all $\geqslant 0$ so that there is no trouble with the square root. Thus, one has a powerful device to avoid computation of small integrals. In an actual calculation, the Schwartz prescreening is not used on the level of individual basis functions but on the level of shell batches because integrals are always calculated in batches. To realize this, the largest exchange integral of a given exchange integral block is looked for and its square root is stored in the so called *pre-screening* matrix **K** (that is stored on disk in ORCA). In a `Direct` SCF this matrix is not recalculated in every cycle, but simply read from disk whenever it is needed. The matrix of exchange integrals

on the level of individual basis function is used in `Conventional` calculations to estimate the disk requirements (the "statistics" run).

Once it has been determined that a given integral batch survives it may be calculated as:

$$\langle ij\,|kl\,\rangle = \sum_p d_{pi} \sum_q d_{qj} \sum_r d_{kr} \sum_s d_{sl}\,\langle i_p j_q\,|k_r l_s\,\rangle \tag{2.85}$$

where the sums p, q, r, s run over the primitive Gaussians in each basis function i, j, k, l and the d's are the contraction coefficients. There are more powerful algorithms than this one and they are also used in ORCA. However, if many terms in the sum can be skipped and the total angular momentum is low, it is still worthwhile to compute contracted integrals in this straightforward way. In equation (2.85), each primitive integral batch $\langle i_p j_q\,|k_r l_s\,\rangle$ contains a prefactor $I_{Gaussians}$ that depends on the position of the four Gaussians and their orbital exponents. Since a contracted Gaussian usually has orbital exponents over a rather wide range, it is clear that many of these primitive integral batches will contribute negligibly to the final integral values. In order to reduce the overhead, the parameter `TCut` is introduced. If the common prefactor I_{pqrs} is smaller than `TCut`, the primitive integral batch is skipped. However, I_{pqrs} is *not* a rigorous upper bound to the true value of the primitive integral. Thus, one has to be more conservative with `TCut` than with `Thresh`. In practice it appears that choosing `TCut=0.01*Thresh` provides sufficient accuracy, but the user is encouraged to determine the influence of `TCut` if it is suspected that the accuracy reached in the integrals is not sufficient.

Hint

- If the direct SCF calculation is close to convergence but fails to finally converge, this maybe related to a numerical problem with the Fock matrix update procedure – the accumulated numerical noise from the update procedure prevents sharp convergence. In this case, set `Thresh` and `TCut` lower and/or let the calculation more frequently reset the Fock matrix (`DirectResetFreq`).

Note

- For a `Direct` calculation, there is no way to have `Thresh` larger than `TolE`. If the errors in the Fock matrix are larger than the requested convergence of the energy, the change in energy can never reach `TolE`. The program checks for that.
- The actual disk space used for *all* temporary files may easily be larger than `MaxDisk`. `MaxDisk` only pertains to the two-electron integral files. Other disk requirements are not currently checked by the program and appear to be uncritical.

2.14.1 Compression and Storage

The data compression and storage options deserve some comment: in a number of modules including RI-MP2, MDCI, CIS, (D) correction to CIS, etc. the program uses so called "Matrix Containers". This means that the data to be processed is stored in terms of matrices in files and is accessed by a double label. A typical example is the exchange operator $\mathbf{K^{ij}}$ with matrix elements $K^{ij}(a,b) = (ia|jb)$. Here the indices i and j refer to occupied orbitals of the reference state and a and b are empty orbitals of the reference state. Data of this kind may become quite large (formally N^4 scaling). To store the numbers in single precision cuts down the memory requirements by a factor of two with (usually very) slight loss in precision. For larger systems one may also gain advantages by also compressing the data (e.g. use a "packed" storage format on disk). This option leads to additional packing/unpacking work and adds some overhead. The simple keywords to control this behavior are given in Table 2.58. For small molecules `UCDOUBLE` is probably the best option, while for larger molecules `UCFLOAT` or particularly `CFLOAT` may be the best choice. Compression does not necessarily slow the calculation down for larger systems since the total I/O load may be substantially reduced and thus (since CPU is much faster than disk) the work of packing and unpacking takes less time than to read much larger files (the packing may reduce disk requirements for larger systems by approximately a factor of 4 but it has not been extensively tested so far). There are many factors contributing to the overall wall clock time in such cases including the total system load. It may thus require some experimentation to find out with which set of options the program runs fastest with.

Caution

- It is possible that `FLOAT` may lead to unacceptable errors. Thus it is not the recommended option when MP2 or RI-MP2 gradients or relaxed densities are computed. For this reason the default is DOUBLE.
- If you have convinced yourself that `FLOAT` is OK, it may save you a factor of two in both storage and CPU.

2.14.2 Distance Dependent Pre-Screening

The "Direct" SCF procedure consists of re-calculating the electron-electron repulsion integrals in every SCF iteration. Avoiding to store these integrals is what made SCF calculations on large molecules feasible and therefore is a cornerstone of modern quantum chemistry. The recalculation is only feasible in reasonable turnaround times of the calculation of negligibly small integrals is avoided before calculation. Thus, there needs to be a reasonably cheap to compute estimate for these integrals that programs rely on for avoiding the actual integral calculation. This is the important subject of pre-screening. Pre-screening reduces the number of integrals to be calculated from O(N^4) (exact integrals) or O(N^3) (RI integrals) to O(N^2) because in a large molecule, there are only O(N) significant charge distributions. Of course, it is beneficial if the integral estimate provides a rigorous upper bound to the actual integral because in this case it is guaranteed that no contribution to the Fock or Kohn-Sham matrix that exceeds a pre-defined accuracy threshold is being missed.

The by far most popular integral estimate is the Schwartz estimator introduced by Häser and Ahlrichs:

$$|(\mu\nu|\kappa\tau)|_{Schwartz} \leq \sqrt{(\mu\nu|\mu\nu)}\sqrt{(\kappa\tau|\kappa\tau)}$$

Thus, all that is required in order to compute this estimate is the pre-screening matrix

$$Q_{\mu\nu} = \sqrt{(\mu\nu|\mu\nu)}$$

Which is cheap and not memory consuming. We note in passing that, since integrals are calculated in batches involving four shells, the pre-screening matrix is also shell contracted, meaning the maximum element over the members of both shells is taken as actual pre-screening element. The extension to RI integrals is trivial and will therefore not be written our here.

The Schwartz estimate is used in practically every quantum chemistry program. Despite it's success, it also has shortcomings. These shortcomings are best understood by considering that a given charge distribution $\mu\nu$ has a multipole structure. At sufficient distance, the interaction of the distributions $\mu\nu$ and $\kappa\tau$ therefore have a distance dependence that depends on the multipolar structure of the two distributions. For example, the monopole-monopole interaction goes as $R-1$ (R being the distance between the centers of the two distributions), dipole-dipole interaction fall off as R^{-3} etc. Evidently, there is no distance dependence in the Schwartz estimate and consequently, it will have a tendency to strongly overestimate integrals that involve non-negligible but distant charge distributions.

Fig. 2.2: Demonstration how the Schwartz-estimate can drastically overestimate integrals. The red trace Is the Schwartz-estimate while the black trace is the actual interaction of two one-center fs charge distributions as a function of distance.

Historically, a different, related estimator has been popular in the early days of direct SCF. This is the overlap estimator (OVLR-estimate). Consider the two-electron integral:

$$(\mu\nu|\kappa\tau) = \int\int \frac{\mu(\mathbf{r}_1)\nu(\mathbf{r}_1)\kappa(\mathbf{r}_2)\tau(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|} d\mathbf{r}_1 d\mathbf{r}_2$$

And pretend that we can replace the inverse electronic distance by an effective distance between the two charge distributions:

$$|\mathbf{r}_1 - \mathbf{r}_2| \rightarrow R_{eff}$$

In this case, the integral simply becomes:

$$(\mu\nu|\kappa\tau) = \frac{S_{\mu\nu}S_{\kappa\tau}}{R_{eff}}$$

With S being the overlap integral. The estimator is then obtained by taking the maximum absolute element of the overlap matrix elements over the members of the involved shells.

$$|(\mu\nu|\kappa\tau)|_{OVLR} = \frac{\widetilde{S}_{\mu\nu}\widetilde{S}_{\kappa\tau}}{R_{eff}}$$

The effective distance is:

$$R_{eff} = R_{\mu\nu,\kappa\tau} - ext_{\mu\nu} - ext_{\mu\nu,\kappa\tau}$$

The estimator is applied if

$$R_{eff} > 1$$

And defaults back to the Schwartz estimate otherwise. In this equation $R_{\mu\nu,\kappa\tau} = |R_{\mu\nu} - R_{\kappa\tau}|$ is the distance between the centers of the two charge distributions ($R_{\mu\nu} = \langle \mu \vee r \vee \nu \rangle$) and $ext_{\mu\nu}$ is the extent of a charge distribution which is discussed in detail in the section on multipole approximations. In a nutshell $ext_{\mu\nu}$ defines a radius outside of which the product $\mu(r)\nu(r)$ is zero. Thus, the condition $R_{eff} > 1$ is the multipole permissible criterion. If it is met, the charge distributions are non-overlapping and the multipole estimate is convergent.

Based on this realization, Ochsenfeld and co-workers have developed a series of integral estimators that are designed to take this distance dependence into account. In their initial work, Lambrecht and Ochsenfeld provided a concise analysis of the multipolar structure of the charge distribution. After some experimentation, they settled on proposing **the QQR-estimator** which is a good compromise in terms of computational cost and efficiency in eliminating small contributions:

$$|(\mu\nu|\kappa\tau)|_{QQR} = \frac{Q_{\mu\nu}Q_{\kappa\tau}}{R_{eff}}$$

Which is applied if $R_{eff} > 1$, as above. This estimate is not completely rigorous, but still very safe because it assumes monopole moments on the two charge distributions. The Schwartz-integral instead of the overlap integral is assumed to mimic the multipole structure of the charge distributions.

Since experience indicates that the QQR estimator is not eliminating significantly more integrals than the Schwartz-estimate, Thomson and Ochenseld proposed a refined estimator that they refer to as "CSAM". It is given by:

$$|(\mu\nu|\kappa\tau)|_{CSAM} = Q_{\mu\nu}Q_{\kappa\tau} \max\left(T_{\mu\kappa}T_{\nu\tau}, T_{\mu\tau}T_{\nu\kappa}\right)$$

Where the T-matrix elements are shell maxima of the two-center Coulomb pre-screening integrals:

$$T_{\mu\kappa} = \frac{\sqrt{(\mu\mu|\kappa\kappa)}}{\sqrt{Q_{\mu\mu}Q_{\kappa\kappa}}}$$

The T-factors bring in the distance dependence of the estimated integral.

Finally, there is a multipole estimator that takes the lowest multipole moment of each charge distribution into account.

$$|(\mu\nu|\kappa\tau)|_{multipole} = max\left(\frac{M^{\mu\nu}_{Lmin,m}M^{\kappa\tau}_{Lmin',m'}}{R^{Lmin+Lmin'+1}}\right)$$

Some idea about the tightness of the estimators can be seen from the following plot:

Fig. 2.3: Tightness of various integral estimators for the model system (glycine)$_3$ with the def2-SVP basis set.

One can see that the Schwartz-estimator is rigorous and the QQR estimator is nearly rigorous but not very much tighter. The OVLR estimator appears to be the by far tightest estimate but is scattering evenly above and below the correct value. CSAM leads to some underestimation of the integral values and appears to be slightly tighter than QQR. The lowest multipole estimate is ridiculous and of unusable quality.

Whether the revised estimators leads to actual computational savings can be seen in a model calculation on (glycine)$_{15}$/def2-SVP :

Screening	Energy /Eh	Cycles	Fock time/s
SCHWARTZ	-3175.706 180 549 541	14	3217.049
CSAM	-3175.706 180 551 857	14	2856.666
QQR	-3175.706 180 550 171	14	3037.277
OVLR	-3175.706 180 550 173	14	3078.138
MULTIPOLE	-3175.706 180 551 146	14	3500.262

Fig. 2.4: actual computation times and total Hartree-Fock energies for (glycine)$_{15}$ under the influence of different integral estimators.

The results show that only CSAM leads to savings on the order of 10%, although at the loss of some accuracy which in this example amounts to about 0.01 micro-Eh. The origin of the limited savings are certainly related to the fact that the lowest multipole of a two-center charge distribution is equal to the overlap integral between the two distributions. Thus, as soon as two basis functions have an overlap, they will have a monopole moment and the decay of the interaction between two such charge distributions is only R^-1^ which falls off so slowly that it is not of practical relevance in the elimination of small integrals.

The following input triggers the different estimates:

```
%shark Prescreening Schwartz
 OVLR
 QQR
 CSAM
 Multipole
End
```

Note

- In practice OVLR and Multipole are highly unstable and are not recommended
- The integral estimates are NOT yet available for RI integrals.

Relevant Papers:

1. Häser, Marco; Ahlrichs, Reinhart. Improvements on the direct SCF method. *Journal of Computational Chemistry*, **1989**, 10 (1), 104–111. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.540100111, DOI: 10.1002/jcc.540100111.
2. Lambrecht, Daniel S.; Ochsenfeld, Christian. Multipole-based integral estimates for the rigorous description of distance dependence in two-electron integrals. *The Journal of Chemical Physics*, **2005**, 123 (18), 184101. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.2079967/15373755/184101_1_online.pdf, DOI: 10.1063/1.2079967.
3. Thompson, Travis H.; Ochsenfeld, Christian. Distance-including rigorous upper bounds and tight estimates for two-electron integrals over long- and short-range operators. *The Journal of Chemical Physics*, **2017**, 147 (14), 144101. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.4994190/14837196/144101_1_online.pdf, DOI: 10.1063/1.4994190.

2.14.3 The "BubblePole" Approximation

The algorithms for Fock matrix construction in ORCA have been optimized to the point where the COSX approximation scales linearly with system size and so does the XC integration. Thus, asymptotically the quadratic scaling of the Coulomb term is dominating the Fock matrix construction time in very large systems. It is well-known that the introduction of hierarchical multipole approximations to the integrals can bring down the scaling of the Coulomb term to linear or near linear. This is the subject of the fast multipole method (FMM) approximation. The FMM approximation has been introduced in ORCA for the calculation of the point charge contribution to embedding calculations and significantly brings down the execution time.

While it may seem tempting to also introduce the FMM approximation in the quantum-quantum charge interactions, the ORCA developers have taken a different route. In FMM algorithms, real space is divided into hierarchical boxes followed by the calculation of the multipole interaction between boxes provided that a multipole allowedness criterion is met.

While it has been proven to work by several authors, it appeared to us that the boxing algorithm is not very natural to chemistry and leads to a number of problems that we intended to avoid in our alternative development.

The Bubblepole (BUPO) approximation is based on a different partitioning that is based on spheres ("Bubbles"). These bubbles fully enclose collections of "quantum objects". These quantum objects may be shell-pairs, auxiliary basis function shells or also point charges. The criterion for grouping a number of such objects together is spatial proximity. Since shell pairs and auxiliary shells are charge distributions, they have a spatial extent which must be taken into account in the group assembly algorithm. Taking the example of shell pairs, each surviving shell is assigned a shell pair center $R_{\mu\nu}$ and an extent $ext_{\mu\nu}$. The precise definition of these quantities is unique to ORCA and are fully discussed in the literature.

The BUPO algorithm then uses a variant of the Kmeans algorithm to group a predefined number of objects (e.g. 150) into each "bottom level" bubble. The bubble center is the arithmetic mean of all enclosed objects and the bubble radius is adjusted such that all objects are fully enclosed together with their extents. This ensures that there is no "leakage" of probability density outside of each bubble.

After setting up the bottom level bubbles, a bubble hierarchy is created in which multipoles are translated from the one bubble layer to the next. Super-bubbles contain lower-level bubbles (typically around three), until at the top layer, there only is a single bubble that encloses the entire molecule. This only happens one initially during system setup).

In the present ORCA implementation, the BUPO algorithm has been combined with ORCA's most efficient Coulomb-construction algorithm – the Split-RI-J algorithm. In the resulting RI-BUPO-J algorithm, the near-field is treated with the Split-RI-J method and the far-field by the hierarchical construction. The RI-J method generally contains three significant steps

"Projection" of the density on the auxiliary basis set:

$$g_K = \sum_{\mu\nu} P_{\mu\nu} \left(\mu\nu|K\right)$$

Solution of linear equations to get the aux-basis density:

$$P_K = \sum_L \left(V^{-1}\right)_{KL} g_L$$

Assembly of the Coulomb matrix:

$$J_{\mu\nu} = \sum_K P_K \left(\mu\nu|K\right)$$

The second step is computationally negligible and proceeds very efficienctly via the Cholesky decomposition. It is not linear scaling but about three to four orders of magnitude cheaper than the other steps such that for all treatable system sizes, it is insignificant. In the first step, the multipole approximation is applied to the density in the orbital basis and therefore the bubbles contain shell-pairs from the basis sets and multipoles derived from the molecular density expanded in the basis set. In the third step, the bubbles contain auxiliary basis shell pairs and multipoles derived from the aux-basis density. The subtleties that derive from this construction are discussed in detail in the original publication.

The numerical results indicate that RI-BUPO-J is asymptotically linear scaling with system size. On linear chains, the crossover with the Split-RI-J algorithm occurs at around 30-40 glycine units. In three dimensional systems, it will be significantly later. Hence, a real advantage of the RI-BUPO/J algorithm over Split-RI-J will only occur for very large systems. This is not due to shortcomings of the BUPO construction but is testimony of the exceptional efficiency of Split-RI-J.

Fig. 2.5: Numerical results for the regular RI-J algorithm, Split-RI-J and RI-BUPO-J on linear glycine chains with basis sets of increasing quality.

There is a significant number of options for BUPO. However, we do NOT recommend to change the defaults. This is expert territory.

```
Simple Keyword line:

! RI-BUPO/J

Detailed parameter definitions:

%shark ExtentOpt Ext_SemiNumeric. # default: how to calculate the extents
Ext_analytic
Ext_Numeric
TSphere 1e-15     # shell pair extent cut-off
```

(continues on next page)

(continued from previous page)

```
MP_LAbsMax_BAS 32 # Max L allowed in BAS multipoles
MP_LAbsMin_BAS 0  # Min L allowed in BAS multipoles
MP_Lmax_BAS 10    # Bottom level expansion length
MP_Lincr_BAS 4    # Expansion length increase for
                  # subsequent bubble levels
MP_Rallow_BAS 1.0 # Far field met if effective bubble
                  # distance larger than this (Bohrs)
MP_TScreen_BAS 1e-10  # Multipole elimination threshold
MP_ClusterDim_BAS 150 # Number of objects bottom level bubble
MP_ClusterDim2_BAS 3  # Number of bubbles in super-bubble
                      # in higher level of the hierarchy
MP_NLevels_BAS 15     # Max number of bubble levels in the
                      # hierarchy
# these parameters apply to the first step of RI-BUPO-J
# the same variables exist w.r.t. the auxbasis using the
# postfix \_AUX and apply to the second step of RI-BUPO-J
End
```

Relevant Papers:

1. Colinet, Pauline; Neese, Frank; Helmich-Paris, Benjamin. Improving the Efficiency of Electrostatic Embedding Using the Fast Multipole Method. *J. Comput. Chem.*, **2025**, 46 (1), e27532. DOI: 10.1002/jcc.27532.
2. Neese, Frank; Colinet, Pauline; DeSouza, Bernardo; Helmich-Paris, Benjamin; Wennmohs, Frank; Becker, Ute. The "Bubblepole" (BUPO) Method for Linear-Scaling Coulomb Matrix Construction with or without Density Fitting. *J. Phys. Chem. A*, **2025**, 129 (10), 2618–2637. DOI: 10.1021/acs.jpca.4c07415.
3. White, Christopher A.; Johnson, Benny G.; Gill, Peter M.W.; Head-Gordon, Martin. The continuous fast multipole method. *Chemical Physics Letters*, **1994**, 230 (1), 8–16. DOI: 10.1016/0009-2614(94)01128-1.

2.14.4 Angular Momentum Limits

Energies and gradients in ORCA can now (starting from ORCA 6.1) be performed up to L=10 in the orbital and aux basis sets. Note that the nomenclature changed to the accepted spectroscopic notation:

L=0	1	2	3	4	5	6	7	8	9	10
S	P	D	F	G	H	I	K	L	M	N

Note

- no j-functions! The program will reject it.
- L is reserved for combined s,p shells for historical reasons. If you want to input L=8 functions, simply use the symbol "8" instead of "L"

Here is an example of how to use the angular momentum definition in an example of a user-specified basis set:

```
! RHF cc-pV6Z VeryTightSCF PrintBasis PModel
%basis

# Basis set for element : Ne
NewGTO Ne
S 11
  1 902400.0000000000 0.0000064569
  2 135100.0000000000 0.0000501790
  3 30750.0000000000 0.0002638325
  4 8710.0000000000 0.0011134543
```

(continues on next page)

(continued from previous page)

```
  5 2842.0000000000 0.0040396234
  6 1026.0000000000 0.0130374676
  7 400.1000000000 0.0377405416
  8 165.9000000000 0.0967943419
  9 72.2100000000 0.2108241400
 10 32.6600000000 0.3586501519
 11 15.2200000000 0.3985795167
S 11
  1 902400.0000000000 -0.0000045125
  2 135100.0000000000 -0.0000351554
  3 30750.0000000000 -0.0001851517
  4 8710.0000000000 -0.0007804845
  5 2842.0000000000 -0.0028452422
  6 1026.0000000000 -0.0092079116
  7 400.1000000000 -0.0271417038
  8 165.9000000000 -0.0715447701
  9 72.2100000000 -0.1678190251
 10 32.6600000000 -0.3267206864
 11 15.2200000000 -0.4994236511
S 1
  1 7.1490000000 1.0000000000
S 1
  1 2.9570000000 1.0000000000
S 1
  1 1.3350000000 1.0000000000
S 1
  1 0.5816000000 1.0000000000
S 1
  1 0.2463000000 1.0000000000
P 5
  1 815.6000000000 0.0014608751
  2 193.3000000000 0.0126013201
  3 62.6000000000 0.0668956161
  4 23.6100000000 0.2559896696
  5 9.7620000000 0.7470043852
P 1
  1 4.2810000000 1.0000000000
P 1
  1 1.9150000000 1.0000000000
P 1
  1 0.8476000000 1.0000000000
P 1
  1 0.3660000000 1.0000000000
P 1
  1 0.1510000000 1.0000000000
D 1
  1 13.3170000000 1.0000000000
D 1
  1 5.8030000000 1.0000000000
D 1
  1 2.5290000000 1.0000000000
D 1
  1 1.1020000000 1.0000000000
D 1
  1 0.4800000000 1.0000000000
F 1
  1 10.3560000000 1.0000000000
F 1
  1 4.5380000000 1.0000000000
F 1
  1 1.9890000000 1.0000000000
```

(continues on next page)

(continued from previous page)

```
F 1
  1 0.8710000000 1.0000000000
G 1
  1 8.3450000000 1.0000000000
G 1
  1 3.4170000000 1.0000000000
G 1
  1 1.3990000000 1.0000000000
H 1
  1 6.5190000000 1.0000000000
H 1
  1 2.4470000000 1.0000000000
I 1
  1 4.4890000000 1.0000000000
K 1
  1 3.0000000000 1.0000000000
# This would be L, but we cannot use the letter L
# because this is used for combined S-P shells
  8 1
  1 3.5000000000 1.0000000000
  M 1
1 2.0000000000 1.0000000000
  N 1
1 1.0000000000 1.0000000000
 end;
end

%shark FockFlag Force_SHARK
       end

* xyz 0 1
Ne 0 0 0
*
```

2.14.5 Keywords

Table 2.58: Simple input keywords related to integral handling

Keyword	Description
`Direct`	Selects an integral direct calculation
`Conv`	Selects an integral conventional calculation
`CheapInts`	Use the cheap integral feature in direct SCF calculations
`NoCheapInts`	Turn that feature off
`KeepInts`	Do not delete the integrals from disk after a calculation in conventional mode
`ReadInts`	Read the existing integrals from a previous calculation in conventional mode
`Float`	Set storage format for numbers to single precision (SCF, RI-MP2, CIS, CIS(D), MDCI)
`Double`	Set storage format for numbers to double precision (default)
`UCFloat`	Use float storage in the matrix containers without data compression
`CFloat`	Use float storage in the matrix containers with data compression
`UCDouble`	Use double storage in the matrix containers without data compression
`CDouble`	Use double storage in the matrix containers with data compression

2.15 The SHARK Integral Package and Task Driver

2.15.1 Preface

Starting with ORCA 5.0 very large changes have taken place in the way that the program handles integrals and integral related tasks like building Fock matrices. SHARK is a powerful and efficient infrastructure that greatly facilitates the handling of these tasks. This allows developers to write highly streamlined code with optimal performance and a high degree of reliability. Compared to the way ORCA handled integrals before ORCA 5.0, tens of thousands of lines of codes, often duplicated or nearly duplicated from closely related parts of the program could be eliminated. From the perspective of the user, the visible changes to the input and output of the program compared to ORCA 4.2.1 and earlier are relatively limited. However, under the hood, the changes are vast and massive and will ensure that ORCA's infrastructure is modern and very well suited for the future of scientific computing.

The benefits of SHARK for the users of ORCA are:

1. Improved code efficiency that is consistent through all program tasks. In particular, complicated two-electron integrals, for example in the context of GIAOs, two-electron spin-orbit coupling and two-electron spin-spin coupling integrals are handled with vastly improved efficiency. Also, integral digestion has been vastly improved with very large benefits for calculations that build many Fock matrices at a time, for example in CIS/TD-DFT, analytic Hessians or response property calculations.
2. Improved code reliability, since all integrals now run through a well debugged, common interface
3. Shorter development times. The new infrastructure is so user friendly to programmers that writing new code that makes use of SHARK is much faster than in the past.
4. SHARK handles basis sets much better than the old infrastructure. Whether the basis sets used follow a segmented contraction, general contraction or partial general contraction is immaterial since the algorithms have been optimized carefully for each kind of basis throughout.

2.15.2 The SHARK integral algorithm

One cornerstone of SHARK is a new integral algorithm that allows for highly efficient evaluation of molecular integrals. The algorithm is based on the beautiful McMurchie-Davidson algorithm which leads to the following equation for a given two-electron integral:

$$(\mu_A\nu_B|\kappa_C\tau_D) = C\sum_{tuv} E_t^{\mu\nu;x} E_u^{\mu\nu;y} E_v^{\mu\nu;z} \sum_{t'u'v'} E_{t'}^{\kappa\tau;x} E_{u'}^{\kappa\tau;y} E_{v'}^{\kappa\tau;z} (-1)^{t'+u'+v'} R_{t+t',u+u',v+v'}$$

Here

$$C = 8\pi^{5/2} = 139.9473466209989027701 03$$

and the primitive Cartesian Gaussian basis functions $\{\mu_A\}$ where A is the atomic center, where basis function μ is centered at position $\mathbf{R}_A$. In order to catch a glimpse of what the McMurchie-Davidson algorithm is about, consider two unnormalized, primitive Gaussians centered at atoms A and B, respectively:

$$G_A = x_A^i y_A^j z_A^k \exp(-\alpha R_A^2)$$
$$G_B = x_B^{i'} y_B^{j'} z_B^{k'} \exp(-\beta R_B^2)$$

By means of the Gaussian product theorem, the two exponentials are straightforwardly rewritten as:

$$\exp(-\alpha R_A^2)\exp(-\beta R_B^2) = K_{AB}\exp\Big(-(\alpha+\beta)r_P^2\Big)$$

With

$$K_{AB} = \exp\left(-\tfrac{\alpha\beta}{\alpha+\beta}|\mathbf{R}_A - \mathbf{R}_B|^2\right)$$

$r_P^2 = |\mathbf{r} - \mathbf{R}_P|^2$ is the electronic position relative to the point

$$\mathbf{R}_P = \tfrac{\alpha}{\alpha+\beta}\mathbf{R}_A + \tfrac{\beta}{\alpha+\beta}\mathbf{R}_B$$

at which the new Gaussian is centered. The ingenious invention of McMurchie and Davidson was to realize that the complicated polynomial that arises from multiplying the two primitive Cartesian Gaussians can be nicely written in terms of Hermite polynomials $\{\Lambda\}$. In one dimension:

$$x_A^i x_B^{i'} = \sum_{t=0}^{i+i'} E_t$$

And hence:

$$G_A G_B = K_{AB} \sum_{t=0}^{i+i'} E_t^{AB} \sum_{u=0}^{j+j'} E_u^{AB} \sum_{v=0}^{k+k'} E_v^{AB} \Lambda_{tuv}^{AB}$$

With

$$\Lambda_{tuv}^{AB} = \left(\frac{\partial}{\partial X_P}\right)^t \left(\frac{\partial}{\partial Y_P}\right)^u \left(\frac{\partial}{\partial Z_P}\right)^v \exp\Big(-(\alpha+\beta)R_P^2\Big)$$

This means that the original four center integral is reduced to a sum of two-center integrals over Hermite Gaussian functions. These integrals are denoted as

$$R_{t+t',u+u',v+v'} = \int\int \Lambda_{tuv}^{AB}(\mathbf{r}_1;\mathbf{R}_P)\Lambda_{t'u'v'}^{CD}(\mathbf{r}_2;\mathbf{R}_Q) r_{12}^{-1} d\mathbf{r}_1 d\mathbf{r}_2$$

With these definitions one understands the McMurchie Davidson algorithm as consisting of three steps:

1. Transformation of the Bra function product into the Hermite Gaussian Basis
2. Transformation of the Ket function product into the Hermite Gaussian Basis
3. Calculation of the Hermite Gaussian electron repulsion integral

SHARK is the realization that these three steps can be efficiently executed by a triple matrix product:

$$(\mu_A \nu_B | \kappa_C \tau_D) = \left(\mathbf{E}^{\text{bra}} \mathbf{R} \mathbf{E}^{\text{ket}}\right)_{\mu\nu,\kappa\tau}$$

Here $\mathbf{E}^{\text{bra}}$ and $\mathbf{E}^{\text{ket}}$ collect the E coefficients for all members of the shell product on the bra and ket side ($E_{\mu\nu,tuv}^{\text{bra}}$ and $E_{\kappa\tau,tuv}^{\text{ket}}$), respectively, and $\mathbf{R}$ collects the integrals over Hermite Gaussian functions ($R_{tuv,t'u'v'}$).

There are many benefits to this formulation:

1. The integral is factorized allowing steps to be performed independent of each other. For example, the E matrices can be calculated at the beginning of the calculation and reused whenever needed. Their storage is unproblematic
2. Matrix multiplications lead to extremely efficient formation of the target integrals and drive the hardware at peak performance
3. Steps like contraction of primitive integrals and transformation from the Cartesian to the spherical Harmonics basis can be folded into the definition of the E matrices thus leading to extremely efficient code with next to no overhead creates by short loops.
4. Programming integrals becomes very easy and efficient. Other types of integrals as well as derivative integrals are readily approached in the same way. Also, two- and three-index repulsion integrals, as needed for the RI approximation are also readily formulated in this way.
5. One-electron integrals are equally readily done with this approach.

There is a very large number of technicalities that we will not describe in this manual which is only intended to provide the gist of the algorithm.

2.15.3 SHARK and libint

Up to ORCA 4.2.1, ORCA has almost entirely relied on the libint2 integral library which is known to be very efficient and powerful. Starting from ORCA 5.0, both SHARK and libint are used for integral evaluations and libint is fully integrated into the SHARK programming environment. Integrals that are only available in one of the packages are done with this package (e. g. GIAO, SOC and Spin-Spin integrals in SHARK; F12 or second derivative integrals in libint). For the integrals available in both packages, the program makes a judicious choice about the most efficient route. The reason for this hybrid approach is the following:

The SHARK integral algorithm is at its best for higher angular momentum functions ($l > 2$; d-functions) which is where the efficiency of the matrix multiplications leads to very large computational benefits. Integrals over, say, four f- or g-functions perform much faster (up to a factor of five) than with traditional integral algorithms. However, for low angular momenta, there is overhead created by the matrix multiplications and also by the fact that the McMurchie Davidson algorithm is known to not be the most FLOP count efficient algorithm. To some extent, this is take care of by using highly streamlined routines for low angular momenta that perform extremely well. However, there are penalties for intermediate angular momenta, where the efficiency of the matrix multiplications has not set in and the integrals are too complicated for hand coding. These integrals perform best with libint and consequently, the program will, by default, select libint to perform such integral batches.

2.15.4 Basis set types

One significant aspect of molecular integral evaluation is the type of contraction that is present in a Gaussian basis set. The most general type of basis set is met in the "general contraction" scheme. Here all primitive Gaussian basis functions of a given angular momentum are collected in a vector $\{\phi\}$. In general, all primitives will contribute to all basis functions $\{\varphi\}$ of this same angular momentum. Hence, we can write:

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_{N_l} \end{pmatrix} = \begin{pmatrix} d_{11} & d_{11} & \cdots & d_{1M_l} \\ d_{21} & d_{21} & \cdots & d_{2M_l} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N_l1} & d_{N_l2} & \cdots & d_{N_lM_l} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{M_l} \end{pmatrix}$$

Where N_l and M_l are the number of actual basis functions and primitives respectively. Typically, the number of primitives is much larger than the number of basis functions. The matrix **d** collects the contraction coefficients for each angular momentum. Typical basis sets that follow this contraction pattern are atomic natural orbital (ANO) basis sets. They are typically based on large primitive sets of Gaussians. Such basis sets put very demands on the integral package since there are many integrals over primitive Gaussian basis functions that need to be generated. If the integral package does not take advantage of the general contraction, then this integral evaluation will be highly redundant since identical integrals will be calculated N_l times (and hence, integrals over four generally contracted shells will be redundantly generated N_l^4 times). SHARK takes full advantage of general contraction for all one- and two-electron integrals that it can generate. Here, the unique advantages of the integral factorization come to full benefit since all integral quadruples of a given atom quadruple/angular momentum quadruple can be efficiently generated by just two large matrix multiplications.

The opposite of general contraction is met with segmented contraction. Here each basis function involves a number of primitives:

$$\varphi_\mu = \sum_k d_{k\mu}\phi_k$$

Quite typically, none of the ϕ_k that occur in the contraction of one basis functions occurs in any other basis function. Typical basis sets of this form are the "def2" basis sets of the Karlsruhe group. They are readily handled by most integral packages and both SHARK and libint are efficient in this case.

The third class of basis sets is met, when general contraction is combined with segmented contraction. Basis sets of this type are, for example, the correlation consistent (cc) basis sets. We call such basis sets "partially generally contracted". In such basis sets, part of the basis functions are generally contracted (for example, the s- and p-functions in main group elements), while other basis functions (e. g. polarization functions, diffuse functions, core correlation functions) are not generally contracted. It is difficult to take full advantage of such basis sets given their complicated structure. In ORCA 5, special code has been provided that transforms the basis set into an intermediate basis set that does not contain any redundancies and hence drives SHARK or libint at peak performance.

In assessing the efficiency vs the accuracy of different integral algorithms, it is clear that segmented basis sets lead to the highest possible efficiency if they are well constructed. For such basis set the pre-screening that is an essential step of any integral direct algorithm performs best. The highest possible accuracy (per basis functions) is met with generally contracted basis sets. However, here the pre-screening becomes rather inefficient since it can only be performed at the level of atom/angular momentum combinations rather than individual shell quadruples. Thus, as soon as a given atom/angular momentum combination leads to any non-negligible integral, all integrals for this combination need to be calculated. This created a sizeable overhead. Consequently, SCF calculations can never be as efficient as with segmented basis sets. If this is immaterial, for example, because a subsequent coupled cluster or MRCI calculation is dominating the calculation time, general contraction is very worthwhile to be explored. For partial general contraction, our algorithm performs very nearly as efficiently as for segmented contraction in SCF calculations. However, since the intermediate basis set is larger than the original orbital basis, certain limited performance penalties can arise in some job types.

2.15.5 Task drivers

In traditional algorithms, quantum chemical programs frequently contain many instances of nested loops over basis function shells, the integral package is called and the integrals are "digested" for a given task. While these steps are inevitable, programming them repeatedly is laborious and error prone. In addition, improvements, say in the handling of contractions or symmetry, need to implemented in many different places. In the SHARK infrastructure all of this is unnecessary since it is programmed in an object-oriented fashion, where the programmer does not need to take care of any detail. Hence, developers only need to write short code sections that distribute the generated integrals into whatever data structure they need, while the SHARK interface takes care of all technical aspects and triggers the sophisticated and efficient machinery that underlies it.

Given this situation, the future of ORCA will involve SHARK taking care of nearly of the compute intensive, laborious tasks, while ORCA will organize and trigger all of these tasks. ORCA and SHARK communicate via a lean and well-defined interface to exchange the necessary data. In this way, a modern, efficient, easy to use and readily maintainable development environment is created.

2.15.6 SHARK User Interface

While SHARK is a large and complicated machinery, we have deliberately kept the interface as straightforward and simple as possible.

In the simple input line there are keywords to turn SHARK on (default) or off:

```
! UseShark
! NoUseShark
```

Deprecated since version 6.0.0: The legacy (non-SHARK code) was removed from the program, so SHARK can no longer be disabled in calculations, which require integrals (as opposed to MM calculations, for example).

Further, there are a few flags that can be set in the `%shark` block:

```
%shark
 UseGeneralContraction false  # turns general contraction algorithm on or
                              # off. There normally is no need to set this
                              # flag since the program will find the
                              # contraction case automatically
 Printlevel 1                 # Amount of output generated. Choose 0 to
                              # suppress output and 2 for more output.
                              # Everything else is debug level printing and
                              # will fill your harddrive very quickly with
                              # unusable information
 PartialGCFlag -1             # Let the program decide whether to use PGC
               0              # do not use it
               1              # Enforce PGC (even for ANO bases)
 FockFlag SHARK_libint_hybrid # default: best of both worlds
          force_shark         # Force Shark where possible
```

(continues on next page)

(continued from previous page)

```
          force_libint        # Force libint where possible
 RIJFlag  RIJ_Auto            # default: program decides the best way
          Split_rij           # new SHARK Split-RI-J algorithm
          Split_rij_2003      # Highly efficient re-implementation of the
                              # Original 2003 algorithm. Mostly used!
          rij_regular         # Use traditional 3 center integrals
                              # (not recommended)
end
```

2.16 SCF Stability Analysis

The SCF stability will give an indication whether the SCF solution is at a local minimum or a saddle point.[173, 174] It is available for RHF/RKS and UHF/UKS. In the latter case, the SCF is restarted by default using new unrestricted start orbitals if an instability was detected. For a demonstration, consider the following input:

```
! BHLYP def2-SVP NORI

%scf
 guess hcore
 HFTyp UHF
 STABPerform true
end

* xyz 0 1
h 0.0 0.0 0.0
h 0.0 0.0 1.4
*
```

The HCORE guess leads to a symmetric/restricted guess, which does not yield the unrestricted solution. The same is often true for other guess options. For more details on the stability analysis, see Section *SCF Stability Analysis*.

The SCF stability analysis evaluates the electronic Hessian (with respect to orbital rotations) at the point indicated by the SCF solution to determine the lowest eigenvalues of the Hessian. If one or more negative eigenvalues are found, the SCF solution corresponds to a saddle point and not a true local minimum in the space considered in the analysis. A typical case are stretched bonds of diatomics, where the symmetry of the initial guess leads to a restricted solution instead of the often preferred unrestricted one. Several spaces are theoretically possible[173]. However, ORCA limits itself to the analysis RHF/RKS in the space of UHF/UKS or UHF/UKS in the space of UHF/UKS. As such, it is on the available for the SCF parts of DFT and HF.[174] In the following, HF is used to indicate both HF and KS. Consider the following input (unless indicated otherwise, default values are shown):

```
! BHLYP def2-SVP NORI

%scf
 guess hcore # for illustrative purposes only
 HFTyp UHF # default based on spin multiplicity
 STABPerform true # default false
 STABRestartUHFifUnstable true # restart the UHF-SCF if unstable
 STABNRoots 3                  # number of eigenpairs sought
 STABMaxDim 3                  # Davidson expansion space = MaxDim * NRoots
 STABMaxIter 100               # maximum number of Davidson iterations
 STABNGuess 4096               # size of initial guess matrix: 4096 x 4096
 STABDTol 0.0001               # convergence criterion from iteration to iteration
 STABRTol 0.0001               # convergence criterion max residual norm
 STABlambda +0.5               # mixing parameter
 STABORBWIN -1, -1, -1, -1,  -1, -1, -1, -1 # defines the donor / acceptor spaces
                               # 4 parameters for RHF
                               # 8 paramters for UHF (4 alpha, 4 beta)
                               # orbital window, -1 refers to automatic␣
```

(continues on next page)

(continued from previous page)

```
↪determination
 STABEWIN -5.0, 5.0            # lower and upper cutoff in Eh for automatic␣
↪freezing
 #-----------------------------------------------------------------------------
 # alternative specification using a sub-block:
 stab
  NRoots 3
  MaxDim 3 # etc.
 end
end

* xyz 0 1
h 0.0 0.0 0.0
h 0.0 0.0 1.4
*
```

The determination of the electronic Hessian is structurally comparable to the TDHF/CIS/TDDFT procedure. Thus, many options are very similar and the user is encouraged to read the section on TDDFT (Section *Excited States via RPA, CIS, TD-DFT and SF-TDA*) to clarify some of the options given here. Since one is usually only interested in the qualitative determination "stable or not?", three roots should be sufficient to find the lowest eigenvalue. By the same philosophy, `StabMaxDim`, `StabMaxIter`, `StabNGuess` and the convergence criteria were chosen. The parameter `StabLambda` refers to the λ of equation 37 of reference [173], which determines the mixing of the original SCF solution and the new orbitals to yield a new guess. Choosing this value is not trivial, since positive and negative values can lead to different new solutions (at least in principle). The convergence of the ensuing SCF depends on it, as well, since all SCF procedures require a sufficiently good guess to converge in a decent number of iterations (or even at all).

The orbital window and the energy window can be specified. Note that the `StabEWIN` will be overridden by the appropriate `StabORBWIN` values. The automatic determination is also influenced by the `%method FrozenCore` settings. Tests have shown that significant curtailing of the actual orbital window can drastically influence the results to the point of qualitative failure.

Current limitations on the method are:

- Only single-point-like calculations are supported. For geometry optimizations etc., one must use the guess MORead feature *Choice of Initial Guess and Restart of SCF Calculations* to employ the guess obtained here. Likewise, one must extract a geometry and run a separate calculation if one is interested in the SCF stability.
- As for TDDFT, NORI, RIJONX, and RIJCOSX are supported. RI-JK is not supported.
- Other, more advanced features like finite-temperature calculations and relativistic calculations (beside ECPs) are not possible at this time.

Overall, the user is cautioned against using the stability analysis blindly without critically evaluating the result in terms of energy difference and by investigating the orbitals (by the printout or by plotting). Its usefulness cannot be denied, but it is certainly not black-box.

An SCF stability analysis with default settings can be requested via STABILITY, SCFSTABILITY, SCFSTAB or STAB on the simple input line.

2.17 Finite Electric Fields

Electric fields can have significant influences on the electronic structure of molecules. In general, when an electric field is applied to a molecule, the electron cloud of the molecule will polarize along the direction of the field. The redistribution of charges across the molecule will then influence the wavefunction of the molecule. Even when polarization effects are not significant, the electric field still exerts a drag on the negatively and positively charged atoms of the molecule in opposite directions, and therefore affect the orientation and structure of the molecule. The combination of electrostatic and polarization effects make electric fields a useful degree of freedom in tuning e.g. reactivities, molecular structures and spectra [175]. Meanwhile, the energy/dipole moment/quadrupole moment changes of the system in the presence of small dipolar or quadrupolar electric fields are useful for calculating many electric properties of the system via numerical differentiation, including the dipole moment, quadrupole moment, dipole-dipole polarizability, quadrupole-quadrupole polarizability, etc. Such finite difference property calculations can be conveniently done using compound scripts in the ORCA Compound Scripts Repository.

An overview of relevant keywords is given in Table 2.59.

2.17.1 Dipolar Electric Fields

A uniform or equivalently speaking, dipolar electric field can be added to a calculation via the `EField` keyword in the `%scf` block:

```
%scf
  EField 0.1, 0.0, 0.0 # x, y, z components (in au) of the electric field
end
```

Although the keyword is in the `%scf` block, it applies the electric field to all other methods (post-HF methods, multireference methods, TDDFT, etc.) as well. Analytic gradient contributions of the electric field are available for all supported methods that already support analytic gradients, but analytic Hessian contributions are not.

> **Warning**
>
> The electric field functionality is not available for *GFN-xTB* and force field methods (as well as any method that involves GFN-xTB or force fields, e.g. *QM/XTB* and *QM/MM*! Combination with these method will result in an abort.

The sign convention of the electric field is chosen in the following way: suppose that the electric field is generated by a positive charge in the negative z direction, and a negative charge in the positive z direction, then the z component of the electric field is positive. This convention is consistent with most but not all other programs [175], so care must be taken when comparing the results of ORCA with other programs.

Another important aspect is the gauge origin of the electric field. The gauge origin of the electric field is the point (or more accurately speaking, one of the points - as there are infinitely many such points) where the electric potential due to the electric field is zero. Different choices of the gauge origin do not affect the geometry and wavefunction of the molecule, as they do not change the electric field felt by the molecule, but they do change the energy of the molecule. The default gauge origin is the (0,0,0) point of the Cartesian coordinate system, but it is possible to choose other gauge origins via the `EFieldOrigin` keyword in the `%scf` block.

```
%scf
 EFieldOrigin CenterOfMass      # use center of mass
              CenterOfNucCharge # use center of nuclear charge
              0.0, 0.0, 0.0     # use given X,Y,Z as origin (default: 0,0,0)
                                # in the units chosen for the coordinates␣
↪(Angstrom/Bohr)
end
```

Note the default gauge origin of the electric field is different from the default gauge origin of the ELPROP module, which is the center of mass. If the user chooses the center of mass/nuclear charge as the gauge origin of the electric field, the gauge origin will move as the molecule translates; this has important consequences. For example, in an MD

simulation of a charged molecule in an electric field, the molecule will not accelerate, unlike when `EFieldOrigin` is fixed at a given set of coordinates, where the molecule will accelerate forever. In general, `CenterOfMass` and `CenterOfNucCharge` are mostly suited for the finite difference calculation of electric properties, where one frequently wants to choose the center of mass or nuclear charge as the gauge origin of the resulting multipole moment or polarizability tensor. Instead, a fixed origin is expected to be more useful for simulating the changes of wavefunction, geometry, reactivity, spectra etc. under an externally applied electric field, as experimentally the electric field is usually applied in the lab frame, rather than the comoving frame of the molecule.

2.17.2 Quadrupolar Electric Fields

Similar to dipolar electric fields, quadrupolar fields can be added via the `QField` keyword in the `%scf` block:

```
%scf
  QField 0.1, 0.0, 0.0, 0.05, 0.0, 0.0 # xx, yy, zz, xy, xz, yz components (in au)
                                       # of the quadrupolar field
end
```

The gauge origin of the quadrupolar field is the same as that of the *dipolar electric field*.

2.17.3 Combination of Dipolar and Quadrupolar Electric Fields

Dipolar and *quadrupolar electric fields* can be combined using the respective `EField` and `QField` keywords in the `%scf` block. This allows one to simulate a gradually varying electric field, for example the following input specifies an electric field that has a strength of 0.01 au at the gauge origin ((0,0,0) by default), pointing to the positive z direction, and increases by 0.001 au for every Bohr as one goes in the positive z direction:

```
%scf
  EField 0.0, 0.0, 0.01
  QField 0.0, 0.0, 0.001, 0.0, 0.0, 0.0
end
```

As a second example, one can also simulate an ion trap:

```
%scf
  QField -0.01, -0.01, -0.01, 0.0, 0.0, 0.0
end
```

Under this quadrupolar field setting, a particle will feel an electric field that points towards the gauge origin, whose strength (in au) is 0.01 times the distance to the gauge origin (in Bohr). This will keep cations close to the origin, but pushes anions away from the origin. Unfortunately, there is no analytic gradient available for quadrupolar fields.

Notes on Electric Fields

- An au (atomic unit) is a fairly large unit for electric fields: 1 au = 51.4 V/Angstrom. By comparison, charged residues in proteins, as well as scanning tunneling microscope (STM) tips, typically generate electric fields within about 1 V/Angstrom; electrode surfaces usually generate electric fields within 0.1 V/Angstrom under typical electrolysis conditions [175]. If the molecule is not close to the source of the electric field, it is even harder to generate strong electric fields: for example, a 100 V voltage across two metal plates that are 1 mm apart generates an electric field of merely 10^{-5} V/Angstrom. Therefore, if experimentally a certain strength of homogeneous electric field seems to promote a reaction, but no such effect is found in calculation, please consider the possibility that the experimentally observed reactivity is due to a strong local electric field near the electrode surface (that is much higher than the average field strength in the system), or due to other effects such as electrolysis. Conversely, if you predict a certain molecular property change at an electric field strength of, e.g. > 0.1 au, it may be a non-trivial question whether such an electric field can be easily realized experimentally.

- The electric field breaks the rotational symmetry of the molecule, in the sense that rotating the molecule can change its energy. Therefore, geometry optimizations in electric fields cannot be done with internal coordinates. When the user requests geometry optimization, the program automatically switches to Cartesian coordinates if it detects an electric field. While Cartesian coordinates allow the correct treatment of molecular rotation, they generally lead to poor convergence, so a large number of iterations is frequently necessary. Also, since certain geometry optimization tasks (notably transition state optimization) can currently only be done under internal coordinates, electric fields cannot be used in such types of optimization tasks yet.
- Similarly, when the molecule is charged, its energy is not invariant with respect to translations. However, when there is only a dipolar electric field but no other translational symmetry-breaking forces (quadrupolar field, point charges, wall potentials), a charged molecule will accelerate forever in the field, and its position will never converge. Therefore, for geometry optimizations within a purely dipolar electric field and no wall potentials, we do not allow global translations of the molecule, even when translation can reduce its energy. For MD simulations we however do allow the global translations of the molecule by default. If this is not desired, one can fix the center of mass in the MD run using the `CenterCOM` keyword (section *Run*).
- For *frequency calculations* in electric fields, we do not project out the translational and rotational contributions of the Hessian (equivalent to setting `ProjectTR false` in `%freq`; see *Vibrational Frequencies* for details). Therefore, the frequencies of translational and rotational modes can be different from zero, and can mix with the vibrational modes. When the electric field is extremely small but not zero, the "true" translational/rotational symmetry breaking of the Hessian may be smaller than the symmetry breaking due to numerical error; this must be kept in mind when comparing the frequency results under small electric fields versus under zero electric field (in the latter case `ProjectTR` is by default true). Besides, when the translational and rotational frequencies exceed `CutOffFreq` (which is 1 cm^{-1} by default; see section *Vibrational Frequencies*), their thermochemical contributions are calculated as if they are vibrations.
- While the program allows the combination of electric fields with an *implicit solvation model*, the results must be interpreted with caution, because the solvent medium does not feel the electric field. The results may therefore differ substantially from those given by experimental setups where both the solute and the solvent are subjected to the electric field. If the solvent's response to the electric field is important, one should use an explicit solvation model instead. Alternatively, one can also simulate the electric field in the implicit solvent by adding inert ions (e.g. Na^+, Cl^-) to the system. Similarly, implicit solvation models cannot describe the formation of electrical double layers in the electric field and their influence on solute properties, so in case electrical double layers are important, *MD simulations* with explicit treatment of the ions must be carried out.
- The electric field not only contributes to the core Hamiltonian, but has extra contributions in GIAO calculations, due to the magnetic field derivatives of dipole integrals. In the case of a dipolar electric field, the GIAO contributions have been implemented, making it possible to study e.g. the effect of electric fields on *NMR shieldings*, and as a special case, nucleus independent chemical shieldings (*NICSs*), which are useful tools for analyzing aromaticity. *Quadrupolar fields* cannot be used in GIAO calculations at the moment.

2.17.4 Keywords

Table 2.59: `%scf` block input keywords and options for finite electric fields.

Keyword	Option	Description
`EField`	`<x, y, z>`	Activates *dipolar electric field* with x, y, z components (in au)
`QField`	`<xx, yy, zz, xz, xz, yz>`	Activates *quadrupolar electric field* with xx, yy, zz, xz, xz, yz components (in au)
`EFieldOri`	`CenterOfMass`	Sets origin to center of mass
	`CenterOfNucCharge`	Sets origin to center of nuclear charge
	`<X, Y, Z>`	Sets origin to X,Y,Z coordinates (default: 0,0,0) in the units chosen for the coordinates (Angstrom/Bohr)

2.18 Fragment Specification

Atoms in a calculation can be grouped into specific *fragments*, which serve multiple purposes. Fragment definitions can be used to *assign Basis Sets and ECPs*, organize output in the *population analysis section*, and enable features like *fragment constrain optimization* and *Rigid Body Optimization*. They are also used in *local energy decomposition* and *multi-level calculations*.

2.18.1 Fragments defined on Input File

There are three ways to assign atoms to fragments using the input file. The first method is to assign a specific atom to a specific fragment by placing `(n)` directly after the atomic symbol in the coordinates section.

```
*xyz -2 2
 Cu(1)  0.00  0.00  0.00
 Cl(2)  2.25  0.00  0.00
 Cl(2) -2.25  0.00  0.00
 Cl(2)  0.00  2.25  0.00
 Cl(2)  0.00 -2.25  0.00
*
```

In this example the fragment feature is used to divide the molecule into a "metal" and a "ligand" fragment and consequently the program will print the metal and ligand charges and populations.

```
---------------------------------------------
CARTESIAN COORDINATES OF FRAGMENTS (ANGSTROEM)
---------------------------------------------

 FRAGMENT 1
  Cu    0.000000    0.000000    0.000000

 FRAGMENT 2
  Cl    2.250000    0.000000    0.000000
  Cl   -2.250000    0.000000    0.000000
  Cl    0.000000    2.250000    0.000000
  Cl    0.000000   -2.250000    0.000000

...

---------------------------------------------
MULLIKEN FRAGMENT CHARGES AND SPIN POPULATIONS
---------------------------------------------
 Fragment   0 :   0.752589     0.842580
 Fragment   1 :  -2.752589     0.157420
Sum of fragment charges         :   -2.0000000
Sum of fragment spin populations:    1.0000000

...

--------------------------------------------
LOEWDIN FRAGMENT CHARGES AND SPIN POPULATONS
--------------------------------------------
 Fragment   0 :   0.222028     0.851552
 Fragment   1 :  -2.222028     0.148448
```

Alternatively, the `%coords` block can be used for fragment definitions in the same way—by placing `(n)` directly after the atomic symbol.

```
%coords
 CTyp   xyz  # the type of coordinates xyz or internal
 Charge -2   # the total charge of the molecule
```

(continues on next page)

(continued from previous page)

```
 Mult   2    # the multiplicity = 2S+1
 coords
    Cu(1)  0.00  0.00  0.00
    Cl(2)  2.25  0.00  0.00
    Cl(2) -2.25  0.00  0.00
    Cl(2)  0.00  2.25  0.00
    Cl(2)  0.00 -2.25  0.00
 end
end
```

Important

- In cases where all atoms are explicitly assigned to fragments, the fragment numbering must start at 1 and use consecutive integers. Non-consecutive or incorrect numbering may lead to errors.
- If any atom is left unassigned (or explicitly assigned to fragment 0), it will be automatically assigned to a fragment using the fragmentation procedure described in *Automatic Fragmentation* section. In such cases—where only a subset of atoms is manually assigned to fragments—it is not necessary to use consecutive fragment numbers. However, the highest fragment number specified in the input must be less than the total number of fragments generated by the combination of manual and automatic procedures. If this condition is not met, ORCA will automatically reorder all fragment numbers in ascending order, starting from 1.

Finally, a third way to define fragments consists of using a `Definition` inside the `%frag` block. In this scheme, the fragment number comes first, followed by a list of atoms (enumerated starting from 0) enclosed in curly brackets `{}` and finishing with `end`. Consecutive atoms can also be specified using the notation `initial_atom:final_atom`.

```
*xyz -2 2
 Cu  0.00  0.00  0.00
 Cl  2.25  0.00  0.00
 Cl -2.25  0.00  0.00
 Cl  0.00  2.25  0.00
 Cl  0.00 -2.25  0.00
*

%frag
 Definition
  1 {0} end    # atom 0 for fragment 1
  2 {1:4} end  # atoms 1 to 4 for fragment 2
 end
end
```

```
*xyz -2 2
 Cl  2.25  0.00  0.00
 Cl -2.25  0.00  0.00
 Cu  0.00  0.00  0.00
 Cl  0.00  2.25  0.00
 Cl  0.00 -2.25  0.00
*

%frag
 Definition
  1 {2} end        # atom 2 for fragment 1
  2 {0:1 3:4} end  # atoms 0, 1, 3, and 4 for fragment 2
 end
end
```

Note

- With the last option (`Definition`) the `%frag` block has to be written after the coordinate section.
- `%frag Definition` also works with coordinates that are defined via an external file.

2.18.2 Automatic Fragmentation

Starting with ORCA 6.1, a set of automatic fragmentation algorithms has been introduced to recognize and group atoms into fragments automatically.

The automatic fragmentation procedure is triggered by including a `%frag` block in the input file or when only a subset of atoms has been manually assigned to fragments.

Automatic fragmentation is performed using a set of procedures that can be selected by the user within the `%frag` block. Each procedure attempts to identify new fragments among atoms that have not yet been assigned. For example, consider a case below where the first 11 atoms belong to a propane molecule and the last three belong to a water molecule. The `Water` procedure in `FragProc` identifies the last three atoms as a water molecule and assigns them to the first fragment, while the `FunctionalGroups` procedure detects and assigns the CH3 and CH2 groups of propane as fragments 2 to 4.

```
%frag
   FragProc Water, FunctionalGroups
end

* xyz 0 1
 C     0.000000     1.270000    -0.260000
 C     0.000000    -0.000000     0.580000
 C     0.000000    -1.270000    -0.260000
 H    -0.890000     1.320000    -0.910000
 H     0.890000     1.320000    -0.910000
 H     0.000000     2.180000     0.370000
 H     0.880000    -0.000000     1.250000
 H    -0.880000    -0.000000     1.250000
 H     0.890000    -1.320000    -0.910000
 H     0.000000    -2.180000     0.370000
 H    -0.880000    -1.300000    -0.910000
 H    -0.920000     0.850000    -2.430000
 O    -1.690000     0.830000    -3.000000
 H    -1.640000     1.630000    -3.510000
*
```

```
  ---------------------------------------------
  CARTESIAN COORDINATES OF FRAGMENTS (ANGSTROEM)
  ---------------------------------------------

   FRAGMENT 1
    H    -0.920000    0.850000   -2.430000
    O    -1.690000    0.830000   -3.000000
    H    -1.640000    1.630000   -3.510000

   FRAGMENT 2
    C     0.000000    1.270000   -0.260000
    H    -0.890000    1.320000   -0.910000
    H     0.890000    1.320000   -0.910000
    H     0.000000    2.180000    0.370000

   FRAGMENT 3
    C     0.000000   -1.270000   -0.260000
    H     0.890000   -1.320000   -0.910000
```

(continues on next page)

(continued from previous page)

```
    H     0.000000   -2.180000    0.370000
    H    -0.880000   -1.300000   -0.910000

   FRAGMENT 4
    C     0.000000   -0.000000    0.580000
    H     0.880000   -0.000000    1.250000
    H    -0.880000   -0.000000    1.250000
```

This automatic fragmentation yields the same result as the manual fragment definition shown below, without the need to inspect the geometry and assign fragments manually.

```
%frag
 Definition
 1 { 11:13} end  # water
 2 { 0 3:5} end  # CH3
 3 { 2 8:10} end # CH3
 4 { 1 6:7} end  # CH2
 end
end
```

Note

- Any fragment defined in the *input file* take precedence over automatic assignments.
- ORCA supports up to 10 procedures in `FragProc`, with the full list provided in Table 2.60 and Table 2.61.
- *Constrained Fragments* do not enable automatically the Automatic Fragmentation when there are atoms unassigned to fragments. However, automatic fragmentation can be activated by including a `%frag` block in the input file.

ORCA provides over 30 `FragProc` methods, which can be combined in a list to generate fragments for various purposes. Below are explanations and examples of the different procedures.

Automatic Fragmentation: Connectivity

`FragProc Connectivity` groups atoms that are connected by chemical bonds, estimated based on atomic radii. In the example below, the first nine atoms belong to a dimethyl ether molecule, which is automatically detected and assigned to one fragment, while the last three atoms—belonging to a water molecule—are assigned to a second fragment.

```
%frag
  PrintLevel 3
  FragProc Connectivity
end

*xyz 0 1
 O     0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 C     1.300000     0.000000    -0.460000
 H    -0.500000     0.870000     1.740000
 H    -0.500000    -0.870000     1.740000
 H     1.000000     0.000000     1.740000
 H     1.300000     0.000000    -1.530000
 H     1.800000     0.870000    -0.100000
 H     1.800000    -0.870000    -0.100000
 H    -1.840000     0.000000    -0.650000
 O    -2.440000     0.000000     0.080000
```

(continues on next page)

(continued from previous page)

```
 H    -3.300000     0.000000    -0.300000
*
```

In this case, the output of the automatic fragmentation tool indicates that two fragments were assigned by the `Connectivity` procedure.

```
----------------------------------------
 Assigned Fragment: 1
----------------------------------------
  Name: Connectivity:0 Method: Connectivity
  Natoms: 9 Charge: 0 Mult: 1

 0 O     0.000000    0.000000    0.000000
 1 C     0.000000    0.000000    1.380000
 2 C     1.300000    0.000000   -0.460000
 3 H    -0.500000    0.870000    1.740000
 4 H    -0.500000   -0.870000    1.740000
 5 H     1.000000    0.000000    1.740000
 6 H     1.300000    0.000000   -1.530000
 7 H     1.800000    0.870000   -0.100000
 8 H     1.800000   -0.870000   -0.100000
----------------------------------------

----------------------------------------
 Assigned Fragment: 2
----------------------------------------
  Name: Connectivity:1 Method: Connectivity
  Natoms: 3 Charge: 0 Mult: 1

 9 H    -1.840000    0.000000   -0.650000
 10 O    -2.440000    0.000000    0.080000
 11 H    -3.300000    0.000000   -0.300000
----------------------------------------
```

Each fragmentation procedure applies only to atoms that have not already been assigned to a fragment. Consequently, connectivity-based fragmentation will ignore any bonds to atoms that have been assigned in the input file or by a previous `FragProc`.

For example, if in the previous case example the oxygen atom is manually assigned to fragment 1 in the input file, this manual assignment takes precedence over the `FragProc Connectivity`. As a result, four fragments are created: one fragment containing the manually assigned oxygen atom, two CH3 groups, and one water molecule.

```
%frag
  PrintLevel 3
  FragProc Connectivity
end


*xyz 0 1
 O(1)  0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 C     1.300000     0.000000    -0.460000
 H    -0.500000     0.870000     1.740000
 H    -0.500000    -0.870000     1.740000
 H     1.000000     0.000000     1.740000
 H     1.300000     0.000000    -1.530000
 H     1.800000     0.870000    -0.100000
 H     1.800000    -0.870000    -0.100000
 H    -1.840000     0.000000    -0.650000
 O    -2.440000     0.000000     0.080000
 H    -3.300000     0.000000    -0.300000
```

(continues on next page)

(continued from previous page)

```
*
```

The output of the automatic fragmentation tool indicates that the first fragment was assigned by the `Orca_Input` procedure, while the remaining three fragments were assigned by the `Connectivity` procedure.

```
=====================================================
Tfragmentator: Fragmenting by Orca_input
=====================================================

-----------------------------------------
 Assigned Fragment: 1
-----------------------------------------
  Name: User-defined:0 Method: Orca_input
  Natoms: 1 Charge: 0 Mult: 1 InputId: 1

 0 O     0.000000    0.000000    0.000000
-----------------------------------------

=====================================================
Tfragmentator: Fragmenting by Connectivity
=====================================================

-----------------------------------------
 Assigned Fragment: 2
-----------------------------------------
  Name: Connectivity:1 Method: Connectivity
  Natoms: 4 Charge: 0 Mult: 1

 1 C     0.000000    0.000000    1.380000
 3 H    -0.500000    0.870000    1.740000
 4 H    -0.500000   -0.870000    1.740000
 5 H     1.000000    0.000000    1.740000
-----------------------------------------

-----------------------------------------
 Assigned Fragment: 3
-----------------------------------------
  Name: Connectivity:2 Method: Connectivity
  Natoms: 4 Charge: 0 Mult: 1

 2 C     1.300000    0.000000   -0.460000
 6 H     1.300000    0.000000   -1.530000
 7 H     1.800000    0.870000   -0.100000
 8 H     1.800000   -0.870000   -0.100000
-----------------------------------------

-----------------------------------------
 Assigned Fragment: 4
-----------------------------------------
  Name: Connectivity:3 Method: Connectivity
  Natoms: 3 Charge: 0 Mult: 1

 9 H    -1.840000    0.000000   -0.650000
 10 O    -2.440000    0.000000    0.080000
 11 H    -3.300000    0.000000   -0.300000
-----------------------------------------
```

Automatic Fragmentation: Atomic and NotAssigned

`FragProc Atomic` and `FragProc NotAssigned` are termination procedures. `FragProc Atomic` assigns each previously unassigned atom to its own individual fragment, while `FragProc NotAssigned` assigns all remaining unassigned atoms to a single fragment.

Similar to other `FragProc` methods, the output of the automatic fragmentation tool indicates that the `Atomic` or `Not_assigned` procedure has been used to generate the corresponding fragments.

```
%frag
  PrintLevel 3
  FragProc Atomic
end

*xyz 0 1
 O(1)  0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 C     1.300000     0.000000    -0.460000
 H    -0.500000     0.870000     1.740000
 H    -0.500000    -0.870000     1.740000
 H     1.000000     0.000000     1.740000
 H     1.300000     0.000000    -1.530000
 H     1.800000     0.870000    -0.100000
 H     1.800000    -0.870000    -0.100000
 H    -1.840000     0.000000    -0.650000
 O    -2.440000     0.000000     0.080000
 H    -3.300000     0.000000    -0.300000
*
```

```
=====================================================
Tfragmentator: Fragmenting by Orca_input
=====================================================

-----------------------------------------
 Assigned Fragment: 1
-----------------------------------------
  Name: User-defined:0 Method: Orca_input
  Natoms: 1 Charge: 0 Mult: 1 InputId: 1

 0 O     0.000000    0.000000    0.000000
-----------------------------------------

=====================================================
Tfragmentator: Fragmenting by Atomic
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 2
-----------------------------------------
  Name: C  Method: Atomic
  Natoms: 1 Charge: 0 Mult: 1

 1 C     0.000000    0.000000    1.380000
-----------------------------------------

-----------------------------------------
 Match: 2, Assigned Fragment: 3
-----------------------------------------
  Name: C  Method: Atomic
  Natoms: 1 Charge: 0 Mult: 1

 2 C     1.300000    0.000000   -0.460000
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------

...

-----------------------------------------
 Match: 11, Assigned Fragment: 12
-----------------------------------------
  Name: H  Method: Atomic
  Natoms: 1 Charge: 0 Mult: 1

 11 H    -3.300000    0.000000   -0.300000
-----------------------------------------
```

```
%frag
  PrintLevel 3
  FragProc NotAssigned
end


*xyz 0 1
 O(1)  0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 C     1.300000     0.000000    -0.460000
 H    -0.500000     0.870000     1.740000
 H    -0.500000    -0.870000     1.740000
 H     1.000000     0.000000     1.740000
 H     1.300000     0.000000    -1.530000
 H     1.800000     0.870000    -0.100000
 H     1.800000    -0.870000    -0.100000
 H    -1.840000     0.000000    -0.650000
 O    -2.440000     0.000000     0.080000
 H    -3.300000     0.000000    -0.300000
*
```

```
====================================================
Tfragmentator: Fragmenting by Orca_input
====================================================

-----------------------------------------
 Assigned Fragment: 1
-----------------------------------------
  Name: User-defined:0 Method: Orca_input
  Natoms: 1 Charge: 0 Mult: 1 InputId: 1

 0 O     0.000000    0.000000    0.000000
-----------------------------------------

====================================================
Tfragmentator: Setting not assigned atoms to a fragment
====================================================

-----------------------------------------
 Assigned Fragment: 1
-----------------------------------------
  Name: Not Assigned Method: Not_assigned
  Natoms: 11 Charge: 0 Mult: 1

 1 C     0.000000    0.000000    1.380000
 2 C     1.300000    0.000000   -0.460000
 3 H    -0.500000    0.870000    1.740000
```

(continues on next page)

(continued from previous page)

```
 4 H    -0.500000   -0.870000    1.740000
 5 H     1.000000    0.000000    1.740000
 6 H     1.300000    0.000000   -1.530000
 7 H     1.800000    0.870000   -0.100000
 8 H     1.800000   -0.870000   -0.100000
 9 H    -1.840000    0.000000   -0.650000
 10 O    -2.440000    0.000000    0.080000
 11 H    -3.300000    0.000000   -0.300000
----------------------------------------
```

Note

- `FragProc NotAssigned` is always applied at the end of all `FragProc` procedures to ensure that no atoms remain without a fragment assignment.

Automatic Fragmentation: Internal Libraries

ORCA includes a series of internal libraries containing definitions of many common molecular structures. In all cases, structure recognition is performed using a VF2 subgraph isomorphism algorithm applied to molecular graphs constructed from Cartesian coordinates.

The available fragmentation procedures that make use of internal libraries are listed in Table 2.60.

Table 2.60: Simple input keywords for Fragment detection

Fragment detection Keyword	Description
`Function`	Contains a list of the most common organic functional groups.
`Aminoaci`	Contains a list of all amino acids, including all common protonation states, but excluding zwitterionic forms.
`AABackbo`	Contains fragment definitions for amino acid backbone detection.
`Backbone`	Performs `AABackbone` followed by merging all fragments into a single protein backbone fragment.
`SeqBackb`	Similar to `AABackbone`, but peptide bonds are assigned as separate fragments.
`AASidech`	Contains a list of all amino acid side chains.
`AASCFine`	Contains a detailed list of organic functional groups within amino acid side chains.
`NABackbo`	Contains fragment definitions for DNA/RNA backbones; the phosphate group is assigned to the 3′ position.
`SEQNABac`	Contains fragment definitions for DNA/RNA backbones; the phosphate group is assigned to the 5′ position.
`NABBFine`	Same as `NABackbone`, but further splits the phosphate group.
`Nucleoti`	Contains a list of all nucleic acids.
`NASidech`	Contains a list of all nucleic acid side chains.
`Solvents`	Contains definitions for common solvents: 1-octanol, n-hexane, cyclohexane, toluene, chlorobenzene, tetrahydrofuran, benzene, N,N-dimethylformamide, pyridine, dimethyl sulfoxide, acetone, ethanol, acetonitrile, methanol, chloroform, carbon tetrachloride, dichloromethane, ammonia, and water.
`Water`	Contains definitions of water molecules, faster than `Solvents` if only fragment water molecules.

Similar to other fragmentation procedures, listing multiple libraries in `FragProc` will apply the procedures consecutively. Using the first example listed in *Automatic Fragmentation* (*input*), and including `PrintLevel 3` in the `%frag` block to increase verbosity, the output will indicate the assignment of fragments by the `Water` and `Functional_groups` procedures.

```
=====================================================
Tfragmentator: Fragmenting by Water
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 1
-----------------------------------------
  Name:  WATER Method: Water
  Natoms: 3 Charge: 0 Mult: 1

 11 H    -0.920000    0.850000   -2.430000
 12 O    -1.690000    0.830000   -3.000000
 13 H    -1.640000    1.630000   -3.510000
-----------------------------------------

=====================================================
Tfragmentator: Fragmenting by Functional_groups
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 2
-----------------------------------------
  Name:  CH3 Method: Functional_groups
  Natoms: 4 Charge: 0 Mult: 1

 0 C     0.000000    1.270000   -0.260000
 3 H    -0.890000    1.320000   -0.910000
 4 H     0.890000    1.320000   -0.910000
 5 H     0.000000    2.180000    0.370000
-----------------------------------------

-----------------------------------------
 Match: 2, Assigned Fragment: 3
-----------------------------------------
  Name:  CH3 Method: Functional_groups
  Natoms: 4 Charge: 0 Mult: 1

 2 C     0.000000   -1.270000   -0.260000
 8 H     0.890000   -1.320000   -0.910000
 9 H     0.000000   -2.180000    0.370000
 10 H    -0.880000   -1.300000   -0.910000
-----------------------------------------

-----------------------------------------
 Match: 1, Assigned Fragment: 4
-----------------------------------------
  Name:  CH2 Method: Functional_groups
  Natoms: 3 Charge: 0 Mult: 1

 1 C     0.000000   -0.000000    0.580000
 6 H     0.880000   -0.000000    1.250000
 7 H    -0.880000   -0.000000    1.250000
-----------------------------------------
```

Automatic Fragmentation: External Libraries

The automated fragmentator also allows users to supply `.xyz` files via the `XZYFRAGLIB` variable in `%frag` block, containing geometries that should be recognized as fragments. The `FragProc Extlib` procedure automatically converts each provided geometry into a molecular graph and then applies a VF2 subgraph isomorphism algorithm—just as is done with the internal libraries.

The following example uses definitions of CH3O and CH3 fragments from the file `Mylib.xyz` to identify and generate fragments within the dimethyl ether geometry provided below:

```
%frag
  PrintLevel 3
  FragProc Extlib
  XZYFRAGLIB "Mylib.xyz"
end

*xyz 0 1
 O     0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 C     1.300000     0.000000    -0.460000
 H    -0.500000     0.870000     1.740000
 H    -0.500000    -0.870000     1.740000
 H     1.000000     0.000000     1.740000
 H     1.300000     0.000000    -1.530000
 H     1.800000     0.870000    -0.100000
 H     1.800000    -0.870000    -0.100000
*
```

Where `Mylib.xyz` contains definitions for methyl and methoxy groups.

```
    5
CHARGE 0 MULT 1 NAME CH3O
 O     0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.380000
 H     1.008807     0.000000     1.736663
 H    -0.328435     0.953845     1.736667
 H    -0.794950    -0.621083     1.736667
    4
CHARGE 0 MULT 1 NAME CH3
 C     0.000000     0.000000     0.000000
 H     0.000000     0.000000     1.070000
 H     1.008807     0.000000    -0.356663
 H    -0.328435    -0.953845    -0.356667
```

The result is the assignment of atoms 0, 1, 3, 4, and 5 to a CH3O fragment, with the remaining atoms assigned to a CH3 by the `Ext_lib` procedure.

```
=====================================================
Tfragmentator: Fragmenting by Ext_lib
=====================================================
****
**** There are 2 Ref structures found in file Mylib.xyz
****

-----------------------------------------
 Match: 1, Assigned Fragment: 1
-----------------------------------------
  Name:  CH3O Method: Ext_lib
  Natoms: 5 Charge: 0 Mult: 1

 0 O     0.000000    0.000000    0.000000
 1 C     0.000000    0.000000    1.380000
```

(continues on next page)

(continued from previous page)

```
 3 H    -0.500000    0.870000    1.740000
 4 H    -0.500000   -0.870000    1.740000
 5 H     1.000000    0.000000    1.740000
----------------------------------------


----------------------------------------
 Match: 1, Assigned Fragment: 2
----------------------------------------
  Name:  CH3 Method: Ext_lib
  Natoms: 4 Charge: 0 Mult: 1

 2 C     1.300000    0.000000   -0.460000
 6 H     1.300000    0.000000   -1.530000
 7 H     1.800000    0.870000   -0.100000
 8 H     1.800000   -0.870000   -0.100000
----------------------------------------
```

Note

- `XZYFRAGLIB` allows the inclusion of up to 10 files. Each file may contain multiple fragment definitions; however, each definition must consist of a single, connected molecule—unconnected structures within a single definition are not allowed.
- The format of `Mylib.xyz` follows the standard XYZ file structure, but optionally supports three identifiers: `CHARGE`, `MULT`, and `NAME`, with `NAME` expected to appear last. These identifiers are printed when a fragment is recognized, though they are not currently used in any calculations.
- Subsequent modifications to fragments by other methods (see *Fusebyatoms* and *Extend* below) do not affect the fragment's CHARGE or MULT values in the identifiers.

Important

- ORCA's VF2 subgraph isomorphism algorithm is based solely on atomic connectivity; it does not consider stereochemistry or bond orders.
- Fragment assignment by `FragProc Extlib` follows the order of the files listed in `XZYFRAGLIB` and the order of fragment definitions within each file. Since an atom is excluded from subsequent matching once it has been assigned to a fragment, the order in which fragments are defined in the library is critically important. For example, in the previous *case*, if CH3 is defined before CH3O in `Mylib.xyz`, both CH3 groups in the *input file* will be matched first. This prevents CH3O from being recognized later, even if it is present in the system.
- Atom assignment to fragments follows the order in which atoms appear in the geometry. In the *previous example*, the CH3O fragment could be assigned using either the C(1)-O or C(2)-O bond. However, since C(1) appears first in the geometry, it is selected for the CH3O fragment.

Automatic Fragmentation: Extend

Many fragments in the internal libraries represent incomplete molecular structures. Therefore, in some cases, it is necessary to add additional hydrogen atoms—or, in certain situations, a hydroxyl group—to complete the system. The `FragProc Extend` addresses this by identifying oxygen atoms bonded to carbon atoms in previously assigned fragments, as well as hydrogen atoms bonded to carbon, nitrogen, or oxygen. It then extends the corresponding fragments to include these atoms.

Consider the example below, which consists of a zwitterionic glycine molecule fragmented using an external library that defines a C–C–N fragment. In this case, `FragProc Extend` can be applied to add the missing oxygen and hydrogen atoms, thereby completing the molecular structure.

```
%frag
  PrintLevel 3
  FragProc Extlib, Extend
  XZYFRAGLIB "Mylib.xyz"
end

*xyz 0 1
 N     0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.460000
 C     1.403962     0.000000     2.015868
 O     1.837767     0.962613     2.627089
 O     2.161436    -0.947142     1.883370
 H     0.423874    -0.764689    -0.525339
 H    -0.538292    -0.884247     1.831954
 H    -0.538292     0.884247     1.831954
 H    -0.970478     0.016940    -0.313504
 H     0.499909     0.831989    -0.313504
*
```

Where `Mylib.xyz` in this case is:

```
    3
Name CCN
 C     0.000000     0.000000     0.000000
 C     0.000000     0.000000     1.500000
 N     1.376503     0.000000    -0.486661
```

The result is an initial assignment of the C–C–N fragment, followed by an extension of the fragment to include two oxygen atoms and five hydrogen atoms.

```
-----------------------------------------
 Match: 1, Assigned Fragment: 1
-----------------------------------------
  Name:  CCN Method: Ext_lib
  Natoms: 3 Charge: 0 Mult: 1

 0 N     0.000000    0.000000    0.000000
 1 C     0.000000    0.000000    1.460000
 2 C     1.403962    0.000000    2.015868
-----------------------------------------

=====================================================
Tfragmentator: Extending Fragments
=====================================================
 Extending Fragment 0 with O  (3)
 Extending Fragment 0 with O  (4)
 Extending Fragment 0 with H  (6)
 Extending Fragment 0 with H  (7)
 Extending Fragment 0 with H  (5)
 Extending Fragment 0 with H  (8)
```

(continues on next page)

(continued from previous page)

```
Extending Fragment 0 with H  (9)
```

Important

- `FragProc Extend` attempts to extend any previously assigned fragment.
- `FragProc Extend` does not modify the `Charge` and `Mult` identifiers of fragments.

Automatic Fragmentation: Fusebyatoms

When multiple fragmentation procedures are used in combination, it may be necessary to merge two previously assigned fragments into one. This can be achieved using the `FragProc Fusebyatoms` procedure, which identifies atom pairs that should belong to the same fragment, as specified in `FuseAtomPairs`.

In the example below, the objective is to fragment a propane molecule into a CH3 group and a CH3–CH2 fragment. One way to achieve this is by first applying `FragProc FunctionalGroups`, which fragments the molecule into two CH3 groups and one CH2 group. The CH2 group can then be merged with one of the CH3 groups. This is done by specifying the carbon atoms of the CH2 and one CH3 group in the `FuseAtomPairs` directive: `FuseAtomPairs {0 1} end`.

```
%frag
        printlevel 3
        FragProc FunctionalGroups, Fusebyatoms
        FuseAtomPairs {0 1} end
end

*xyz 0 1
 C     0.000000     1.270000    -0.260000
 C     0.000000    -0.000000     0.580000
 C     0.000000    -1.270000    -0.260000
 H    -0.890000     1.320000    -0.910000
 H     0.890000     1.320000    -0.910000
 H     0.000000     2.180000     0.370000
 H     0.880000    -0.000000     1.250000
 H    -0.880000    -0.000000     1.250000
 H     0.890000    -1.320000    -0.910000
 H     0.000000    -2.180000     0.370000
 H    -0.880000    -1.300000    -0.910000
*
```

The output from the fragmentator first reports the initial fragmentation performed by the `Functional_groups` library, followed by a message indicating the subsequent fusion of fragments as specified by the `FuseAtomPairs` directive.

```
=====================================================
Tfragmentator: Fragmenting by Functional_groups
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 1
-----------------------------------------
  Name:  CH3 Method: Functional_groups
  Natoms: 4 Charge: 0 Mult: 1

 0 C     0.000000    1.270000   -0.260000
 3 H    -0.890000    1.320000   -0.910000
 4 H     0.890000    1.320000   -0.910000
 5 H     0.000000    2.180000    0.370000
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------

-----------------------------------------
 Match: 2, Assigned Fragment: 2
-----------------------------------------
  Name:  CH3 Method: Functional_groups
  Natoms: 4 Charge: 0 Mult: 1

 2 C     0.000000   -1.270000   -0.260000
 8 H     0.890000   -1.320000   -0.910000
 9 H     0.000000   -2.180000    0.370000
 10 H    -0.880000   -1.300000   -0.910000
-----------------------------------------

-----------------------------------------
 Match: 1, Assigned Fragment: 3
-----------------------------------------
  Name:  CH2 Method: Functional_groups
  Natoms: 3 Charge: 0 Mult: 1

 1 C     0.000000   -0.000000    0.580000
 6 H     0.880000   -0.000000    1.250000
 7 H    -0.880000   -0.000000    1.250000
-----------------------------------------

=====================================================
Tfragmentator: Fusing Fragments
=====================================================
Fusing Fragment 0 (atom 0) with Fragment 2 (atom 1)
```

Automatic Fragmentation: Delete and advanced fragmentation workflows

Almost all fragmentation schemes have an associated delete procedure, which removes the fragments generated by that specific scheme. This enables the construction of advanced fragmentation workflows, where a procedure can temporarily "protect" certain atoms from being fragmented by subsequent methods. These protected fragments can later be deleted, allowing the atoms to be reprocessed using a different fragmentation approach.

Consider the system *below*, which consists of both a phenylalanine molecule and a benzene molecule. The goal is to fragment the phenylalanine into its backbone, one CH2 group, and the phenyl ring, while fragmenting the benzene into six individual CH fragments.

On one hand, by combining the three fragmentation procedures — `FragProc AABackbone, Extend, AASCFinegrained` — the desired fragmentation of phenylalanine can be achieved. The `AABackbone` and `Extend` options identify the aminoacid backbone, while `AASCFinegrained` separates the CH2 group and the phenyl ring into distinct fragments. Meanwhile, the benzene molecule can be fragmented into six individual CH fragments using `FragProc Extlib`, which relies on an external library defining the CH fragment. However, these two fragmentation schemes are incompatible. When `FragProc AASCFinegrained` is applied to the entire system, it fragments not only the phenylalanine side chain but also breaks the benzene ring into a phenyl group and a hydrogen atom. Conversely, if `FragProc Extlib` is applied first to benzene, it may inadvertently fragment the phenylalanine residue, leading to undesired structural splits.

A viable approach to achieve the desired fragmentation involves first identifying the amino acid using `FragProc Aminoacids`. This procedure assigns all atoms that belong to amino acid fragments, thereby protecting them from reassignment by subsequent fragmentation steps. The benzene molecule can then be fragmented using an external library via `FragProc Extlib`. Finally, the phenylalanine fragment is removed using `FragProc DelAminoacids`, allowing it to be re-fragmented as needed. All these operations are executed by simply concatenating the procedures as follows: `FragProc Aminoacids, Extlib, DelAminoacids, AABackbone, Extend, AASCFinegrained`

```
%frag
  PrintLevel 3
  FragProc Aminoacids, Extlib, DelAminoacids, AABackbone, Extend, AASCFinegrained
  XZYFRAGLIB "Mylib.xyz"
  STOREFRAGS true
end

*xyz 0 1
 O     1.300780     3.093320     1.571260
 C     0.407700     2.826000     0.770390
 O    -0.101890     3.606620    -0.030470
 C    -0.115350     1.396690     0.770390
 N    -1.564350     1.396690     0.770390
 H    -1.900780     1.872730     1.595200
 H     0.242580     0.884560     1.663540
 H    -1.901490     0.444630     0.770390
 H    -1.901016     1.873069    -0.054118
 C     0.436090     0.696200    -0.461750
 H     1.512450     0.863150    -0.502770
 H    -0.018950     1.150700    -1.341790
 C     0.178410    -0.790680    -0.515340
 C    -0.895760    -1.288470    -1.262580
 C    -1.134680    -2.667040    -1.312270
 C    -0.299410    -3.547820    -0.614730
 C     0.774760    -3.050040     0.132500
 C     1.013680    -1.671470     0.182200
 H     1.842330    -1.287460     0.758630
 H     1.419110    -3.729500     0.670600
 H    -0.483720    -4.611290    -0.653070
 H    -1.963330    -3.051050    -1.888700
 H    -1.540110    -0.609010    -1.800680
 C     1.181325    -3.732146    -2.620584
 C     2.016592    -4.612923    -1.923046
 C     3.090772    -4.115131    -1.175824
 C     3.329684    -2.736563    -1.126139
 C     2.494417    -1.855785    -1.823677
 C     1.420237    -2.353577    -2.570900
 H     0.770518    -1.668458    -3.113485
 H     0.422252    -4.288369    -3.129823
 H     1.830753    -5.685253    -1.961693
 H     3.740491    -4.800250    -0.633239
 H     4.165243    -2.349352    -0.544907
 H     2.680256    -0.783455    -1.785030
*
```

Where “Mylib.xyz” in this case is:

```
2
NAME CH
C 0.00 0.00 0.00
H 1.08 0.00 0.00
```

The complete fragmentation sequence follows the same structure as in previous examples. In this case, the message `Deleted 1 fragments` is printed after the fragment assignment performed by `Ext_lib`, indicating that a fragment previously defined by `FragProc Aminoacids` has been successfully removed.

```
=====================================================
Tfragmentator: Fragmenting by Amino_Acids
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 1
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------
  Name:  CPHE Method: Amino_Acids
  Natoms: 21 Charge: -1 Mult: 1

 0 O     1.300780    3.093320    1.571260
 1 C     0.407700    2.826000    0.770390
 2 O    -0.101890    3.606620   -0.030470
 3 C    -0.115350    1.396690    0.770390
 4 N    -1.564350    1.396690    0.770390
 5 H    -1.900780    1.872730    1.595200
 6 H     0.242580    0.884560    1.663540
 9 C     0.436090    0.696200   -0.461750
 10 H     1.512450    0.863150   -0.502770
 11 H    -0.018950    1.150700   -1.341790
 12 C     0.178410   -0.790680   -0.515340
 13 C    -0.895760   -1.288470   -1.262580
 14 C    -1.134680   -2.667040   -1.312270
 15 C    -0.299410   -3.547820   -0.614730
 16 C     0.774760   -3.050040    0.132500
 17 C     1.013680   -1.671470    0.182200
 18 H     1.842330   -1.287460    0.758630
 19 H     1.419110   -3.729500    0.670600
 20 H    -0.483720   -4.611290   -0.653070
 21 H    -1.963330   -3.051050   -1.888700
 22 H    -1.540110   -0.609010   -1.800680
-----------------------------------------

====================================================
Tfragmentator: Fragmenting by Ext_lib
====================================================
****
**** There are 1 Ref structures found in file Mylib.xyz
****

-----------------------------------------
 Match: 1, Assigned Fragment: 2
-----------------------------------------
  Name:  CH Method: Ext_lib
  Natoms: 2 Charge: 0 Mult: 1

 23 C     1.181325   -3.732146   -2.620584
 30 H     0.422252   -4.288369   -3.129823
-----------------------------------------

-----------------------------------------
 Match: 2, Assigned Fragment: 3
-----------------------------------------
  Name:  CH Method: Ext_lib
  Natoms: 2 Charge: 0 Mult: 1

 24 C     2.016592   -4.612923   -1.923046
 31 H     1.830753   -5.685253   -1.961693
-----------------------------------------

...

-----------------------------------------
 Match: 6, Assigned Fragment: 7
-----------------------------------------
  Name:  CH Method: Ext_lib
  Natoms: 2 Charge: 0 Mult: 1
```

(continues on next page)

(continued from previous page)

```
 28 C     1.420237   -2.353577   -2.570900
 29 H     0.770518   -1.668458   -3.113485
-----------------------------------------

=====================================================
Tfragmentator: Deleting Fragments (Amino_Acids)
=====================================================
 Deleted 1 fragments
=====================================================
Tfragmentator: Fragmenting by Backbone
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 7
-----------------------------------------
  Name: CO-NH3-CH Method: AA_Backbone
  Natoms: 8 Charge: 0 Mult: 0

 0 O     1.300780    3.093320    1.571260
 1 C     0.407700    2.826000    0.770390
 3 C    -0.115350    1.396690    0.770390
 4 N    -1.564350    1.396690    0.770390
 5 H    -1.900780    1.872730    1.595200
 6 H     0.242580    0.884560    1.663540
 7 H    -1.901490    0.444630    0.770390
 8 H    -1.901016    1.873069   -0.054118
-----------------------------------------
=====================================================
Tfragmentator: Extending Fragments
=====================================================
 Extending Fragment 6 with O  (2)

=====================================================
Tfragmentator: Fragmenting by AA_SideChains_FG
=====================================================

-----------------------------------------
 Match: 1, Assigned Fragment: 8
-----------------------------------------
  Name: Ph Method: AA_SideChains_FG
  Natoms: 11 Charge: 0 Mult: 1

 12 C     0.178410   -0.790680   -0.515340
 13 C    -0.895760   -1.288470   -1.262580
 14 C    -1.134680   -2.667040   -1.312270
 15 C    -0.299410   -3.547820   -0.614730
 16 C     0.774760   -3.050040    0.132500
 17 C     1.013680   -1.671470    0.182200
 18 H     1.842330   -1.287460    0.758630
 19 H     1.419110   -3.729500    0.670600
 20 H    -0.483720   -4.611290   -0.653070
 21 H    -1.963330   -3.051050   -1.888700
 22 H    -1.540110   -0.609010   -1.800680
-----------------------------------------

-----------------------------------------
 Match: 1, Assigned Fragment: 9
-----------------------------------------
  Name: CH2 Method: AA_SideChains_FG
  Natoms: 3 Charge: 0 Mult: 1
```

(continues on next page)

(continued from previous page)

```
 9 C     0.436090    0.696200   -0.461750
10 H     1.512450    0.863150   -0.502770
11 H    -0.018950    1.150700   -1.341790
----------------------------------------
```

Table Table 2.61 lists the corresponding delete procedure associated with each fragmentation method.

Table 2.61: Simple input keywords for Fragment detection and their corresponding deletion procedure

Fragment detection Keyword	Fragment deletion Keyword
`Extlib`	`DELExtlib`
`Connectivity`	`DELConnectivity`
`Atomic`	`DELAtomic`
`FunctionalGroups`	`DELFunctionalGroups`
`NotAssigned`	
`Backbone`	`DELBackbone`
`SeqBackbone`	`DELSeqBackbone`
`AABackbone`	`DELAABackbone`
`Aminoacids`	`DELAminoacids`
`AASideChains`	`DELAASideChains`
`AASCFineGrained`	`DELAASCFineGrained`
`NABackbone`	`DELNABackbone`
`NABBFineGrained`	`DELNABBFineGrained`
`SEQNABackbone`	`DELSEQNABackbone`
`NucleoticAcid`	`DELNucleoticAcid`
`NASideChains`	`DELNASideChains`
`Solvents`	`DELSolvents`
`Water`	`DELWater`
`Extend`	
`FuseByAtoms`	

2.18.3 Options available in the `%frag` input block

Table Table 2.62 contains a list of the options available in the `%frag` input block.

Table 2.62: List of options in the `%frag` input block

Option	Type	Default	Description
`Printlevel`	Integer	`1`	Verbose output control for automated fragmentation.
`STOREFRAGS`	Boolean	`False`	Stores assigned fragments in a `.fragments.xyz` file.
`DoInterFragBon`	Boolean	`False`	Automatically detects bonds between fragments for CoVaLED analysis.
`XZYFRAGLIB`	String	`None`	Filenames used in `FragProc Extlib`.
`FragProc`	See Table 2.60, and Table 2.61	`ExtLib, Connectivi`	Fragmentation procedures to be applied automatically.
`Usetopology`	Boolean	`False`	Generate main geometry graph based on .prms file.

continues on next page

Table 2.62 – continued from previous page

Option	Type	Default	Description
`TopolFile`	String	`""`	Topology file name to been used when `Usetopology True`
`PrintInputFlag`	Boolean	`True`	Writes a `%frag` block equivalet to current calculation fragments.

2.19 ORCA and Symmetry

For most of its life, ORCA did not take advantage of molecular symmetry. Starting from version 2.8 (released in September 2010), there has been at least limited use. On request (using the simple keyword `UseSym` for instance, see below), the program detects the point group, orients the molecule, cleans up the coordinates and produces symmetry-adapted molecular orbitals.

Only for geometry cleanup the full point group is taken into account. For all other purposes such as the construction of symmetry-adapted molecular orbitals and or to describe electronic states, only D_{2h} and subgroups are currently supported. Here the use of symmetry helps to control the calculation and the interpretation of the results.

2.19.1 Getting started

Utilization of symmetry is turned on by the simple keyword `UseSymmetry` (which may be abbreviated by `UseSym`), or if a `%Symmetry` (or `%Sym`) input block is present in the input. ORCA will then automatically determine the point group, reorient and center the molecule to align its symmetry elements with the coordinate system, and replace the input structure by a geometry that corresponds exactly to this point group and which minimizes the sum of square distances between the atoms of both structures.

Any program that attempts to find the point group of an arbitrary atom cluster must be prepared to cope with some amount of numerical noise in the atom coordinates. ORCA by default allows each atom to deviate at most 10^{-4} atomic units from the ideal position that is consistent with the point group being examined. The rationale behind this value is the rounding error that occurs when the user feeds Cartesian coordinates with five significant digits after the decimal point into the program which otherwise represent an exact (symmetry-adapted) geometry. A threshold that is about one order of magnitude higher than the numerical noise in the coordinates is usually very safe.

If the maximum error in the Cartesian coordinates exceeds these 10^{-4} atomic units, the symmetry module in ORCA will fail to recognize the expected point group. The user is strongly advised to always make sure that the detected point group meets their expectations. If the point group reported by the symmetry module appears to be too low, the user may try to increase the detection threshold to 10^{-3} or 10^{-2} Bohr radii using option `SymThresh` in the `%Symmetry` input block:

```
%Sym SymThresh 0.01 End
```

A great method to obtain a structure with perfect symmetry avoiding any expensive calculation is to use the simple keywords `! NoIter XYZFile` with an appropriate threshold. The structure in the resulting file with the extension `.xyz` may then be used as input for the actual calculation.

To give an illustrative example, coordinates for staggered ethane have been obtained by geometry optimization *without* using symmetry. If symmetry is turned on, point group C_i is recognized instead of the expected point group D_{3d} due to the remaining numerical noise. To counter this, the detection threshold is increased to 10^{-2} a. u. and a coordinate file with perfect symmetry is produced by the following input:

```
! RHF SVP NoIter XYZfile
%sym SymThresh 1.0e-2 end
*xyz 0 1
  C    -0.002822   -0.005082   -0.001782
  C    -0.723141   -1.252323   -0.511551
  H     0.017157    0.029421    1.100049
```

(continues on next page)

(continued from previous page)

```
  H     1.042121    0.030085   -0.350586
  H    -0.495109    0.917401   -0.350838
  H    -0.743120   -1.286826   -1.613382
  H    -0.230855   -2.174806   -0.162495
  H    -1.768085   -1.287489   -0.162747
*
```

If ORCA fails to find the expected point group even though a value of 10^{-2} atomic units has been selected for `SymThresh`, the user is strongly advised to take a careful look at the structure by means of their favorite visualization tool before increasing this value any further. Look for any obvious distortions or even missing atoms. An especially tricky point may be the orientation of methyl groups or the conformation of floppy side chains. A small rotation about a single bond may be enough to push some atom positions above the limit. If the conformational deviations cannot be fixed using a molecular editor or modelling program, a possible alternative may be to pre-optimize the structure without symmetry using a less expensive method like `PB86` and a small basis set like `def2-SVP`. Even several passes of pre-optimization and structure editing may be considered until all symmetry-equivalent side chains are locked in the same conformation so that ORCA finally detects the correct point group.

It is not recommended to run calculations using a value of `SymThresh` which is much too high or much too small since this may result in some really strange behavior of the symmetry module. Consider for instance the following input file which contains a perfectly octahedral geometry of a sulfur hexafluoride molecule. Its coordinates may be easily created by hand by placing the sulfur atom into the origin and two fluorine atoms on each coordinate axis at equal distances r from the origin ($r = 1.56$ Å or approximately 2.95 atomic units). Using a value for `SymThresh` as large as 0.1 Bohr radii works fine in this case, resulting in the correct point group O_h.

```
# Sulfur hexafluoride (SF6), point group Oh.
! BP86 def2-SVP
%Sym SymThresh 0.1 End
* xyz 0 1
S   0.00   0.00   0.00
F   1.56   0.00   0.00
F  -1.56   0.00   0.00
F   0.00   1.56   0.00
F   0.00  -1.56   0.00
F   0.00   0.00   1.56
F   0.00   0.00  -1.56
*
```

However, if `SymThresh` is increased further to $t = 0.5$ atomic units, the point group detection algorithm breaks down (strange warnings are printed as a consequence) and the reported point group decreases to C_i (in which the center of inversion is the only non-trivial symmetry element). This is because the center of inversion is easy to detect and this is done by one of the early checks. The breakdown of the point group recognition may be explained as follows. During the process of point group detection the symmetry module is of course unaware that the given input geometry is exact. Hence it will be treated as any other input structure. A value of $t = 0.5$ Bohr radii for `SymThresh` means that the unknown exact atom position is located within a sphere of radius $t = 0.5$ atomic units around the input atom position. The input distance $a = \sqrt{2}\,r$ between two adjacent fluorine atoms is approximately $a \approx 2.21$ Å ≈ 4.17 a. u., so their unknown exact distance d may vary in the following interval (see the diagram in Fig. 2.6):

$$d_{\text{min}} = a - 2t = 3.17\,\text{a.u.} \leq d \leq d_{\text{max}} = a + 2t = 5.17\,\text{a.u.}$$

Analogously, the unknown exact distance d' between two opposite fluorine atoms with the input distance $a' = 2r = 5.90$ a. u. is:

$$d'_{\text{min}} = a' - 2t = 4.90\,\text{a.u.} \leq d' \leq d'_{\text{max}} = a' + 2t = 6.90\,\text{a.u.}$$

Since the possible intervals of d and d' overlap (due to $d_{\text{max}} > d'_{\text{min}}$), *all* fifteen F–F distances are considered equal. Since there is no solid with six vertices and fifteen equal inter-vertex distances in three dimensions, the point group detection algorithm fails.

Fig. 2.6: The relation between the value t of `SymThresh`, the distance a of some input atom pair, and the allowed interval $[d_{\min}, d_{\max}]$ for the distance d between the exact atom positions. This interval has the width $d_{\max} - d_{\min} = 4t$.

2.19.2 Geometry optimizations using symmetry

If a geometry optimization is performed with symmetry turned on, ORCA will first determine the point group of the starting structure and replace the geometry that is presumed to contain numerical noise with one that has perfect symmetry. Starting with ORCA 6, the optimizer will clean up the gradient at every step of the optimization if requested by setting option `CleanUpGradient true` in the `%Symmetry` input block. The gradient cleanup is done by projecting out all components that are not totally symmetric. This way the symmetry of the molecule cannot decrease during the optimization.

By default, the point group is determined from scratch again after the geometry has been updated at every step of the optimization. This behaviour may be switched off by setting option `SymRelaxOpt false` in the `%Symmetry` input block. In this case the point group of the molecule is actually frozen during the entire optimization.

The following table summarizes the behaviour of the optimizer depending on the options `SymRelaxOpt` and `CleanUpGradient`:

`SymRelaxOpt`	`CleanUpGradient`	Behaviour
`true`	`true`	Symmetry may increase but not decrease.
`true`	`false`	Symmetry may change freely.
`false`	`true`	Symmetry will be frozen.
`false`	`false`	Setting not recommended.

Setting both switches `false` would allow the point group to change during the optimization but at the same time, a change would be impossible to detect. Therefore this setting is strongly discouraged.

2.19.3 Default alignment of the symmetry elements with the coordinate system

If ORCA determines the point group of a molecule and the user has not selected any special options, the following principles apply to the manner in which the symmetry elements of the full point group are aligned with the coordinate system:

1. The center of mass of the molecule will be shifted into the origin by default.[1] If the point group leaves one *unique* vertex invariant to all symmetry operations, the center of mass agrees with this vertex. This is the case for all point groups except C_s, C_n $(n \geq 1)$, C_{nv} $(n \geq 2)$, and $C_{\infty v}$.
2. If the molecule exhibits a unique axis of symmetry with the highest number of positions, this axis will become the z axis. This applies to all point groups except C_1, C_i, C_s, D_2, D_{2h}, the cubic point groups, and K_h.
3. For point group C_s, the mirror plane will become the xy plane.
4. For point groups C_{nv} $(n \geq 2)$, one of the vertical mirror planes will become the xz plane.
5. For point groups D_n $(n \geq 3)$, D_{nh} $(n \geq 3)$, and D_{nd} $(n \geq 2)$, one of the two-fold rotation axes perpendicular to the axis with the highest number of positions will become the x axis.
6. For point groups D_2, D_{2h}, T, and T_h, the three mutually orthogonal C_2 axes will become the coordinate axes.
7. For point groups T_d, O, and O_h, the three mutually orthogonal four-fold rotation or rotation-reflection axes will become the coordinate axes.
8. Finally, for point groups I and I_h, one of the five sets of three mutually orthogonal C_2 axes will become the coordinate axes. The pair of C_5 or S_{10} axes closest to the z axis will be located in the yz plane.
9. In general the orientation of the molecule will be changed as little as possible to meet the criteria above. If the input geometry meets these criteria already, the molecule will not be moved or rotated at all.

If the point group of the system is D_{nd} with $n \geq 2$ or T_d and the user has selected subgroup C_{2v} using option `PreferC2v`, the following rules apply instead:

- For point group D_{nd} with $n \geq 2$, one of the diagonal mirror planes will become the xz plane.
- For point group T_d, one of the diagonal mirror planes containing the z axis will become the xz plane, i. e. the molecule will be rotated by 45 degrees about the z axis compared to the default orientation.

Table 2.63 gives an overview over all point groups and the way in which the symmetry elements of the reduced point group (the largest common subgroup of D_{2h}) are aligned with the coordinate system.

Table 2.63: Point groups and corresponding subgroups suitable for electronic-structure calculations.

Full point group	Index n	Unique centerPage 195, 2	Consistent with planar moleculePage 195, 3	Chosen subgroup	Alignment of the subgroupPage 195, 4
C_1		no	no	C_1	
C_i		i	no	C_i	
C_s		no	yes	C_s	
C_n	odd	no	no	C_1	
	even	no	no	C_2	z axis
C_{nv}	odd	no	no	C_s	xz plane
	even	no	for $n = 2$	C_{2v}	z, xz, yz
C_{nh}	odd	yes	yes	C_s	xy plane
	even	i	yes	C_{2h}	z, xy
D_n	odd	yes	no	C_2	x axis
	even	yes	no	D_2	
D_{nh}	odd	yes	yes	C_{2v}	x, xy, xz
	even	i	yes	D_{2h}	
D_{nd}	odd	i	no	C_{2h}	x, yz
	even	yes	no	D_2	

continues on next page

[1] In the very special case that the Z matrix contains no atoms with mass, the geometric center will be used instead.

Table 2.63 – continued from previous page

Full point group	Index n	Unique center[2]	Consistent with planar molecule[3]	Chosen subgroup	Alignment of the subgroup[4]
				C_{2v}	z, xz, yz
S_{2n}	odd	i	no	C_i	
	even	yes	no	C_2	z axis
T		yes	no	D_2	
T_h		i	no	D_{2h}	
T_d		yes	no	D_2	
				C_{2v}	z, xz, yz
O		yes	no	D_2	
O_h		i	no	D_{2h}	
I		yes	no	D_2	
I_h		i	no	D_{2h}	
$C_{\infty v}$		no	no	C_{2v}	z, xz, yz
$D_{\infty h}$		i	no	D_{2h}	
K_h		i	no	D_{2h}	

2.19.4 Irreducible representations of D_{2h} and subgroups

Table 2.64, Table 2.65, and Table 2.66 contain lists of the irreducible representations (also called species) and the corresponding characters of the point groups supported for electronic structure calculations in ORCA, and the product tables of these irreducible representations. Where the data depends on the alignment of the symmetry elements with the coordinate system, Mulliken's recommendations [176] are followed. This approach is in line with the recommendations by the IUPAC [177].

Table 2.64: Species and species product table of point group C_{2v}. The species table for C_{2v} corresponds to Table III in [176]. The directions of the two-fold axis and the mirror planes in each column are related to each other by cyclic permutations.

C_{2v}	E	$C_2(z)$ $C_2(x)$ $C_2(y)$	$\sigma_v(xz)$ $\sigma_v(xy)$ $\sigma_v(yz)$	$\sigma_v(yz)$ $\sigma_v(xz)$ $\sigma_v(xy)$
A_1	$+1$	$+1$	$+1$	$+1$
A_2	$+1$	$+1$	-1	-1
B_1	$+1$	-1	$+1$	-1
B_2	$+1$	-1	-1	$+1$

$\times$	A_1	A_2	B_1	B_2
A_1	A_1	A_2	B_1	B_2
A_2	A_2	A_1	B_2	B_1
B_1	B_1	B_2	A_1	A_2
B_2	B_2	B_1	A_2	A_1

Table 2.65: Species and species product table of point group D_2. The species table for D_2 has been obtained by dropping the center of inversion and the mirror planes from the species table for D_{2h} (see Table 2.66).

D_2	E	$C_2(z)$	$C_2(y)$	$C_2(x)$
A	$+1$	$+1$	$+1$	$+1$
B_1	$+1$	$+1$	-1	-1
B_2	$+1$	-1	$+1$	-1
B_3	$+1$	-1	-1	$+1$

$\times$	A	B_1	B_2	B_3
A	A	B_1	B_2	B_3
B_1	B_1	A	B_3	B_2
B_2	B_2	B_3	A	B_1
B_3	B_3	B_2	B_1	A

[2] A center of inversion is denoted i. "yes" indicates the existence of a *unique* vertex that remains invariant to all symmetry operations of the point group.

[3] This column indicates whether the given point group may be the *full* point group of a planar molecule.

[4] This column contains the elements (axes or planes) of the coordinate system that coincide with the symmetry elements of the reduced point group (the largest common subgroup of the full point group and D_{2h}) by *default*. If the full point group contains a unique principle axis of symmetry (with the highest number of positions), this axis is presumed to coincide with the z axis.

Table 2.66: Species and species product table of point group D_{2h}. The species table for D_{2h} corresponds to Table IV in [176].

D_{2h}	E	$C_2(z)$	$C_2(y)$	$C_2(x)$	i	$\sigma(xy)$	$\sigma(xz)$	$\sigma(yz)$
A_g	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
B_{1g}	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1
B_{2g}	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1
B_{3g}	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$
A_u	$+1$	$+1$	$+1$	$+1$	-1	-1	-1	-1
B_{1u}	$+1$	$+1$	-1	-1	-1	-1	$+1$	$+1$
B_{2u}	$+1$	-1	$+1$	-1	-1	$+1$	-1	$+1$
B_{3u}	$+1$	-1	-1	$+1$	-1	$+1$	$+1$	-1

$\times$	A_g	B_{1g}	B_{2g}	B_{3g}	A_u	B_{1u}	B_{2u}	B_{3u}
A_g	A_g	B_{1g}	B_{2g}	B_{3g}	A_u	B_{1u}	B_{2u}	B_{3u}
B_{1g}	B_{1g}	A_g	B_{3g}	B_{2g}	B_{1u}	A_u	B_{3u}	B_{2u}
B_{2g}	B_{2g}	B_{3g}	A_g	B_{1g}	B_{2u}	B_{3u}	A_u	B_{1u}
B_{3g}	B_{3g}	B_{2g}	B_{1g}	A_g	B_{3u}	B_{2u}	B_{1u}	A_u
A_u	A_u	B_{1u}	B_{2u}	B_{3u}	A_g	B_{1g}	B_{2g}	B_{3g}
B_{1u}	B_{1u}	A_u	B_{3u}	B_{2u}	B_{1g}	A_g	B_{3g}	B_{2u}
B_{2u}	B_{2u}	B_{3u}	A_u	B_{1u}	B_{2g}	B_{3g}	A_g	B_{1u}
B_{3u}	B_{3u}	B_{2u}	B_{1u}	A_u	B_{3g}	B_{2g}	B_{1g}	A_g

2.19.5 Options available in the `%Symmetry` input block

Table 2.67 contains a list of the options available in the `%Symmetry` (or `%Sym`) input block. Options `SymThresh` and `SymRelax` (same as `SymRelaxSCF` below) can also be accessed in the `%Method` input block for backward compatibility. This use is deprecated and not recommended in new input files, however.

Table 2.67: List of options in the `%Symmetry` (`%Sym`) input block

Option	Type	Default	Description
`UseSymmetry`	Boolean	`True`	By setting this option to `False`, symmetry may be switched off even though the `%Symmetry` block is present in the input file.
`UseSym`	Boolean	`True`	Same as `UseSymmetry`.
`SymThresh`	Real	10^{-4}	Two vertices with a distance shorter than this threshold (in atomic units) are considered identical during point group recognition.
`PreferC2v`	Boolean	`False`	Indicates whether to prefer subgroup C_{2v} over D_2 for electronic-structure calculations where both choices are appropriate (point groups D_{nd} with odd n and T_d).
`PointGroup`	String	Empty string	If the user specifies a point group using this option, point group recognition will be skipped and the user must make sure that the molecule is oriented in the coordinate system in agreement with the conventions in *Default alignment of the symmetry elements with the coordinate system*. Note that the point group label must be enclosed in double quotes. Otherwise ORCA will complain about an invalid assignment.
`SymRelaxSCF`	Boolean	`False`	Indicates whether orbital occupation numbers of each irreducible representation are allowed to change during SCF.
`SymRelaxOpt`	Boolean	`True`	Indicates whether the point group will be determined from scratch in every step of a geometry optimization. A value of `True` will allow the point group to change in an arbitrary manner. Otherwise the initial point group will be imposed at every step no matter how far the distances between the current and the ideal structure exceed `SymThresh`.

continues on next page

Table 2.67 – continued from previous page

Option	Type	Default	Description
CleanUpCoords	Boolean	True	Determines whether the molecular geometry will be cleaned up using the automatically detected or user-specified point group. Even if CleanUpCoords is False, symmetrized coordinates will still be computed temporarily and a warning will be printed if the largest deviation from the original geometry exceeds SymThresh.
CleanUpGeom	Boolean	True	Same as CleanUpCoords.
CleanUpGrad	Boolean	True	Indicates whether the full point group of the molecule shall be used to remove all non-totally symmetric components from the gradient. This ensures that the point group will not decrease throughout the optimization.
CleanUpGradien	Boolean	True	Same as CleanUpGrad.
Print	Integer	1	Determines the output size for symmetry handling in general and point group detection in particular; 0 – No output during point group detection; 1 – Normal output; 2 – Detailed information; 3 – Debug print.
PrtSALC	Integer	0	Specifies the output size for the construction of symmetry-adapted linear combinations (SALCs) of atomic orbitals; 0 – No output for symmetry-adapted orbitals; 1 – Normal output; 2 – Detailed information (e. g. the SALCs themselves); 3 – Debug print.

2.20 Choice of Initial Guess and Restart of SCF Calculations

The initial guess is an important issue in each SCF calculation. If this guess is reasonable, the convergence of the procedure will be much better. ORCA makes some effort to provide a good initial guess and gives the user enough flexibility to tailor the initial guess to his or her needs.

The initial guess is also controlled via the %scf block and the variables Guess, MOInp and GuessMode.

```
%scf
  Guess     HCore       # One electron matrix
            Hueckel     # Extended H\"uckel guess
            PAtom       # Polarized atomic densities
            PModel      # Model potential
            MORead      # Restart from an earlier calc.
  MOInp     "Name.gbw"  # orbitals used for MORead
  GuessMode FMatrix     # FMatrix projection
            CMatrix     # Corresponding orbital projection
  AutoStart true        # try to use the orbitals from the existing
                        # GBW file of the same name (if possible)
                        # (default)
            false       # don't use orbitals from existing GBW file
end
```

2.20.1 One Electron Matrix Guess

The simplest guess is to diagonalize the one electron matrix to obtain starting orbitals. This guess is very simple but usually also a disaster because it produces orbitals that are far too compact.

2.20.2 Basis Set Projection

The remaining guesses (may) need the projection of initial guess orbitals onto the actual basis set. In ORCA there are two ways this can be done. `GuessMode FMatrix` and `GuessMode CMatrix`. The results from the two methods are usually rather similar. In certain cases `GuessMode CMatrix` may be preferable. `GuessMode FMatrix` is simpler and faster. In short the `FMatrix` projection defines an effective one electron operator:

$$\hat{f} = \sum_p \varepsilon_p a_p^\dagger a_p \tag{2.86}$$

where the sum is over all orbitals of the initial guess orbital set, $a_p^\dagger$ is the creation operator for an electron in guess MO p, a_p is the corresponding annihilation operator and ε_i is the orbital energy. This effective one electron operator is diagonalized in the actual basis and the eigenvectors are the initial guess orbitals in the target basis. For most wavefunctions this produces a fairly reasonable guess.

`CMatrix` is more involved. It uses the theory of corresponding orbitals to fit each MO subspace (occupied, partially occupied or spin-up and spin-down occupied) separately [178, 179]. After fitting the occupied orbitals, the virtual starting orbitals are chosen in the orthogonal complement of the occupied orbitals. In some cases, especially when restarting ROHF calculations, this may be an advantage. Otherwise, it is not expected that `CMatrix` will be grossly superior to `FMatrix` for most cases.

2.20.3 PModel Guess

The PModel guess (chosen by `Guess PModel` in the `%scf` block or simply a keyword line with `!PModel`) is one that is usually considerably successful. It consists of building and diagonalizing a Kohn–Sham matrix with an electron density which consists of the superposition of spherical neutral atoms densities which are predetermined for both relativistic and nonrelativistic methods. This guess is valid for both Hartree–Fock and DFT methods, but not for semiempirical models. However, due to the complexity of the guess it will also take a little computer time (usually less than one SCF iteration). The model densities are available for most atoms of the periodic table and consequently the PModel guess is usually the method of choice (particularly for molecules containing heavy elements) unless you have more accurate starting orbitals available.

2.20.4 Hückel and PAtom Guesses

The extended Hückel guess proceeds by performing a minimal basis extended Hückel calculation and projecting the MOs from this calculation onto the actual basis set using one of the two methods described above. The minimal basis is the STO-3G basis set. The Hückel guess may not be very good because the STO-3G basis set is so poor. There is also accumulating evidence that the superposition of atomic densities produces a fairly good initial guess. The critique of the atomic density method is that the actual shape of the molecule is not taken into account and it is more difficult to reliably define singly occupied orbitals for ROHF calculations or a reasonable spin density for UHF calculations. Therefore ORCA chooses a different way in the `PAtom` guess (which is the default guess): the Hückel calculation is simply carried out *for all electrons* in a minimal basis of atomic SCF orbitals. These were determined once and for all and are stored inside the program. This means that the densities around the atoms are very close to the atomic ones, all orbitals on one center are exactly orthogonal, the initial electron distribution already reflects the molecular shape and there are well defined singly occupied orbitals for ROHF calculations.

2.20.5 Restarting SCF Calculations

To restart SCF calculations, it can be very helpful and time-saving to read in the orbital information of a previous calculation. To do this, specify:

```
! moread
%moinp "name.gbw"
```

This is done by default for single-point calculations if the `.gbw` file of the same name exists – see *AutoStart feature*.

The program stores the current orbitals in every SCF cycle. Should a job crash, it can be restarted from the orbitals that were present at this time by just re-running the calculation to use the present `.gbw` file. In addition, an effort has been made to make `.gbw` files from different releases compatible with each other. If your input `.gbw` file is from an older release, use `! rescue moread noiter` with `% moinp "name.gbw"` to produce an up-to-date `.gbw`. When the `rescue` keyword is invoked, only the orbital coefficients are read from the `.gbw` file, and everything else from the input file. Thus, make sure that the geometry and the basis set of the old `.gbw` file and the new input match.

Within the same ORCA version, neither the geometry nor the basis set stored in `name.gbw` need to match the present geometry or basis set. The program merely checks if the molecules found in the current calculation and `name.gbw` are consistent with each other and then performs one of the possible orbital projections. If the two basis sets are identical the program by default only reorthogonalizes and renormalizes the input orbitals. However, this can be overruled by explicitly specifying `GuessMode` in the `% scf` block as `CMatrix` or `FMatrix`.

If redundant components were removed from the basis (see *Linear Dependence*), then `! moread noiter` must not be used to read SCF orbitals from a previous calculation, as it is going to lead to wrong results. In that case, `! moread` may be used (without `noiter`) if doing the entire calculation in one go is not possible.

If `!moread` and `%moinp` are used, the input `.gbw` file from the earlier calculation must have a different name than the new calculation, because in the very beginning of a calculation, a new `.gbw` file is written. If the names are the same, the `.gbw` file from the earlier calculation is overwritten and all information is lost. Therefore, if you want to restart a calculation with an input file of the same name as the previous calculation, you have to rename the `.gbw` file first. This is good practice anyway to avoid surprises, particularly for expensive calculations. Alternatively, you can use the *AutoStart feature* instead of explicitly specifying `!moread` and `%moinp`.

There is an additional aspect of restarting SCF calculations — if you have chosen `SCFMode = Conventional` the program stores a large number of integrals that might have been time consuming to calculate on disk. Normally the program deletes these integrals at the end of the calculation. However, if you want to do a closely related calculation that requires the same integrals (i.e. the *geometry*, the *basis set* and the *threshold* `Thresh` are the same) it is faster to use the integrals generated previously. This is done by using `KeepInts = true` in the `% scf` block of the first calculation and then use `ReadInts = true` in the `% scf` block of the second calculation. If the second calculation has a different name than the first calculation you have to use `IntName = "FirstName"` to tell the program the name of the integral files. Note that the file containing the integrals does not have an extension — it is simply the name of the previous input file with `.inp` stripped off.

```
%scf
  KeepInts true         # Keep integrals on disk
  ReadInts true         # Read integrals from disk
  IntName  "MyInts"     # Name of the integral files without extension
end
```

Note that, in general, restarting calculations with old integral files requires the awareness and responsibility of the user. If properly used, this feature can save considerable amounts of time.

2.20.6 AutoStart feature

Older versions of ORCA always created a new GBW file at the beginning of the run no matter whether a file of the same name existed or perhaps contained orbitals. Now, in the case of single-point calculations the program automatically checks if a `.gbw` file of the same name exists. If yes, the program checks if it contains orbitals and all other necessary information for a restart. If yes, the variable `Guess` is set to `MORead`. The existing `.gbw` file is renamed to `BaseName.ges` and `MOInp` is set to this filename. If the `AutoStart` feature is not desired, set `AutoStart false` in the `%scf` block or give the keyword `!NoAutoStart` in the simple input line. Note that `AutoStart` is ignored for geometry optimizations: in this case, using previously converged orbitals contained in a `.gbw` file (of a different name) can be achieved via `MORead` and `MOInp`.

2.20.7 Changing the Order of Initial Guess MOs and Breaking the Initial Guess Symmetry

Occasionally you will want to change the order of initial guess MOs — be it because the initial guess yielded an erroneous occupation pattern or because you want to converge to a different electronic state using the orbitals of a previous calculation. Reordering of MOs and other tasks (like breaking the symmetry of the electronic wavefunction) are conveniently handled with the `Rotate` feature in ORCA. `Rotate` is a subblock of the SCF block that allows you to linearly transform pairs of MOs.

```
%scf
  Rotate
   { MO1, MO2, Angle }
   { MO1, MO2, Angle, Operator1, Operator2 }
   { MO1, MO2} # Shortcut to swap MO1 and MO2. Angle=90 degrees.
  end
end
```

Here, `MO1` and `MO2` are the indices of the two MOs of interest. Recall that ORCA starts counting MOs with index 0, i.e. the MO with index 1 is the *second* MO. `Angle` is the rotation angle in degrees. A rotation angle of 90° corresponds to flipping two MOs, an angle of 45° leads to a 50:50 mixture of two MOs, and a 180° rotation leads to a change of phase. `Operator1` and `Operator2` are the orbitals sets for the rotation. For UHF calculations spin-up orbitals belong to operator 0 and spin-down orbitals to operator 1. RHF and ROHF calculations only have a single orbital set.

Among others, the `Rotate` feature can be used to produce broken-symmetry solutions, for example in transition metal dimers. In order to do that, first perform a high-spin calculation, then find the pairs of MOs that are symmetric and antisymmetric combinations of each other. Take these MOs as the initial guess and use rotations of 45° for each pair to localize the starting MOs. If you are lucky and the broken symmetry solution exists, you have a good chance of finding it this way. See section *Broken-Symmetry Wavefunctions and Exchange Couplings* for more details on the broken-symmetry approach.

2.20.8 Automatically Breaking of the Initial Guess Symmetry

Another simple way to break the initial guess symmetry for more trivial cases, is to simply use the keyword `! GUESSMIX`. This will automatically mix 50% of the alpha LUMO into the alpha HOMO. That is equivalent to a 45 degree rotation as done above and only for these orbitals. It might be useful when one wants an open-shell singlet and needs the alpha and beta orbitals to start differently.

The specific angle of rotation can be controlled with:

```
%scf
  guessmix 75 # angle in degrees, default is 45
end
```

2.20.9 Calculating only the energy of an input density

In case you want to give the result of a previous SCF and recalculate the energy, or maybe some other property (like the MP2 energy) using that density without changing the orbitals, you can use the flags `!CALCGUESSENERGY NOITER`.

The SCF program will read the orbitals, compute the density and one Fock matrix necessary to get the energy and move on with no orbital updates. This can be used to combine DFT orbitals with DLPNO-CCSD(T) for example. Be careful with the results you get from this because these orbitals are not variational anymore!

2.20.10 Keywords

Table 2.68: Simple input keywords related to the initial guess.

Keyword	Description
`PAtom`	Selects the polarized atoms guess
`PModel`	Selects the model potential guess
`Hueckel`	Selects the extended Hückel guess
`HCore`	Selects the one-electron matrix guess
`MORead`	Read MOs from a previous calculation (use together with `%moinp "myorbitals.gbw"`)
`AutoStart`	Try to start from an existing GBW file of the same name as the present one (default; only for single-point calculations)
`NoAutoStart`	Don't try to do that
`NoIter`	Sets the number of SCF iterations to 0. This works together with MOREAD and means that the program will work with the provided starting orbitals.
`KeepInts`	Do not delete the integrals from disk after a calculation in conventional mode
`ReadInts`	Read the existing integrals from a previous calculation in conventional mode (use together with `%scf IntName "PreviousBaseName" end`)
`CalcGuessEner`	Calculate the energy of the initial guess orbitals (use together with `NoIter`)

2.21 Frozen Core Options

The frozen core (FC) approximation is usually applied in correlated calculation and consists in neglecting correlation effects for electrons in the low-lying core orbitals. The FC approximation and the number of core electrons per element can be adjusted in the %method block. The default number of core electrons per element is listed in Table 2.69.

Table 2.69: Default values for number of frozen core electrons.

H																	He
0																	0
Li	Be											B	C	N	O	F	Ne
0	0											2	2	2	2	2	2
Na	Mg											Al	Si	P	S	Cl	Ar
2	2											10	10	10	10	10	10
K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
10	10	10	10	10	10	10	10	10	10	10	10	18	18	18	18	18	18
Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe
18	18	28	28	28	28	28	28	28	28	28	28	36	36	36	36	36	36
Cs	Ba	Lu	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn
36	36	46	46	46	46	46	46	46	46	46	46	68	68	68	68	68	68
Fr	Ra	Lr	Rf	Db	Sg	Bh	Hs	Mt	Ds	Rg	Cn						
68	68	68	100	100	100	100	100	100	100	100	100						
Lan-thanide			La	Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb	
			36	36	36	36	36	36	36	36	36	36	36	36	36	36	
Ac-tinides			Ac	Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	
			68	68	68	68	68	68	68	68	68	68	68	68	68	68	

For systems containing heavy elements, core electrons might have higher orbital energies compared to the orbital energies of valence MOs of some lighter elements. In that case, core electrons might be included in the correlation calculation, which ultimately leads to large errors in correlation energy. In order to prevent this, the MO ordering is checked: Do all lower energy MOs in the core region have core electron character, i.e. are they strongly localized on the individual elements? For post-(CAS)SCF calulations, this check is always performed both after the SCF calculation, and after the initial guess (because the SCF may be skipped with `!NoIter`). For other calculations, the check is off by default but may be switched on with the `CheckFrozenCore` keyword in the `%method` block. If core orbitals are found in the valence region, while more delocalized orbitals are found in the core region, the corresponding MO pairs are swapped. This behavior can be disabled using the `CorrectFrozenCore` keyword.

```
%method
  FrozenCore FC_ELECTRONS #Freeze all core electrons
             FC_EWIN      #Freeze selected core electrons via an energy window
                          #e.g. for MP2: %mp2 EWin EMin,EMax
             FC_NONE      #No frozencore approximation
             -n           #Freeze a total of n electrons

  NewNCore Bi 68 end      #Set the number of core electrons for Bi to 68
  CheckFrozenCore   true  #Check whether frozen core orbitals are ordered correctly
                          #Default: true only for post-(CAS)SCF calculations
  CorrectFrozenCore true  #Whether to rotate valence orbitals out of the core␣
↪region
end
```

Note

- The FrozenCore options are applied to all post Hartree-Fock methods.
- If including all electrons is desired, the `!NoFrozenCore` keyword can be simply inserted. For MP2: Frozen virtual orbitals are not allowed in gradient runs or geometry optimization!
- If ECPs are used, the number for NewNCore has to include the electrons represented by the ECPs as well. E.g. if an element is supposed to have 60 electrons in the ECP and additional 8 electrons should be frozen in the correlation calculation, NewNCore should be 68.
- In ORCA we use rather conservative frozencore settings, i. e. a large number of electrons are included in the correlation treatment. Therefore, we recommend to use properly optimized correlating basis functions in all cases, such as the cc-pwCVXZ basis sets.
- For DLPNO calculations the virtual space for core-core and core-valence correlation is adjusted by default, which is described in detail in section *Including (semi)core orbitals in the correlation treatment*.
- In general, `NewNCore` only has an effect in calculations with `FC_ELECTRONS`. In calculations using the DLPNO approximation (except DLPNO-NEVPT2), `NewNCore` has also an effect in the other cases, as is described in section *Including (semi)core orbitals in the correlation treatment*.
- Double-hybrid density functional (section *Double-Hybrid DFT*) calculations by default use the FrozenCore option for the perturbative part, as is the case for MP2.

2.22 CP-SCF Options

The coupled perturbed self-consistent field (CP-SCF) equations have to be solved in many cases, such as when second derivative properties (e.g. vibrational frequencies, polarizability, NMR shielding, indirect spin-spin coupling, hyperfine coupling, g-tensor) or the MP2 relaxed density (in this case they are referred to as Z-vector equations) are calculated. They are a set of linear equations generally expressed as

$$\mathbf{A}\mathbf{U}^x = \mathbf{B}^x,$$

where $\mathbf{U}^x$ is the vector of solutions for perturbation x, the right-hand side (RHS) matrix $\mathbf{B}^x$ is perturbation-specific and the left-hand side (LHS) matrix $\mathbf{A}$ is perturbation-independent and contains, among other terms, the two-electron repulsion integrals $(ij|ab)$ and $(ia|jb)$. The equations are solved iteratively and the LHS is reassembled at every step, while the RHS does not change. The generation and transformation of the two-electron integrals are therefore the most time-consuming parts of the CP-SCF solution.

The ORCA module which solves these equations accepts several options given below with their default values:

```
%method
  Z_Solver     Pople # (default) Use the Pople algorithm to solve the equations
               DIIS  # Use the DIIS algorithm
               CG    # Use the conjugate gradient algorithm
  Z_Tol        1e-3  # Convergence tolerance for the residual norm.
                     # Default is 1e-5 for VeryTightOpt
                     # and varies from 3e-3 to 3e-6 from LooseSCF to ExtremeSCF
  Z_MaxIter    128   # Maximum number of iterations
  Z_MaxDIIS    12    # Maximum number of DIIS vectors
  Z_Shift      0.3   # Level shift for DIIS
  Z_GridXC     1     # XC angular grid used for the LHS
  Z_IntAccXC   3.467 # XC radial grid accuracy used for the LHS
  Z_GridX      1     # COSX angular grid used for the LHS
  Z_IntAccX    3.067 # COSX radial grid accuracy used for the LHS
  Z_GridX_RHS  2     # COSX grid used for the RHS of MP2 Z-vector eqs (see below)
  Z_COSX_Alg   0     # (default) choose the best COSX algorithm automatically
               1     # better prescreening, more efficient for few densities
```

(continues on next page)

(continued from previous page)

```
                2     # uses more memory, more efficient for many densities
end
```

Since ORCA 6, the same settings are used for all electric response property calculations as well as for CIS/TD-DFT gradients and relaxed densities. For convenience, the keywords in the `%elprop` input block are still available but they modify the same internal variables as those in `%method`. For magnetic response properties, the solver and convergence tolerance are set separately in `%eprnmr`, because the convergence behavior of the magnetic response CP-SCF equations is sometimes different.

```
%elprop
  Solver     # Alias, see: %method Z_Solver
  Tol        # Alias, see: %method Z_Tol
  MaxIter    # Alias, see: %method Z_MaxIter
  MaxDIIS    # Alias, see: %method Z_MaxDIIS
  LevelShift # Alias, see: %method Z_Shift
end

%eprnmr
  Solver     # Solver for magnetic response, see options at: %method Z_Solver
  Tol        # Convergence tolerance for magnetic response
  MaxIter    # Alias, see: %method Z_MaxIter
  MaxDIIS    # Alias, see: %method Z_MaxDIIS
  LevelShift # Alias, see: %method Z_Shift
end
```

The keywords `Z_GridX` and `Z_IntAccX` are applicable if the RIJCOSX approximation is chosen for the treatment of two-electron integrals. They determine the angular and radial COSX integration grids, as discussed in section *Changing TD-DFT, CP-SCF and Hessian Grids*. Analogously, the keywords `Z_Grid` and `Z_IntAcc` determine the integration grid for DFT XC functionals.

Integrals on the RHS are evaluated differently for different perturbations - refer to sections *RIJCOSX*, *EPRNMR - keywords for magnetic properties*, *RIJCOSX-RI-MP2 Gradients*, *MP2 and RI-MP2 Second Derivatives* and *RI-MP2 and Double-Hybrid DFT Response Properties* for SCF-level gradients, EPR/NMR calculations with GIAOs, MP2 gradients, and MP2 second derivatives, respectively. For MP2 Z-vector equations, the RIJCOSX Fock-response terms in the RHS are evaluated with the COSX grid specified by `Z_GridX_RHS`. Note that it is used differently to `Z_GridX`: instead, it selects one of the three grids used in the SCF (see Sections *RIJCOSX*, *COSX Numerical Grids*, *Numerical Integration* for details) and it is not recommended to change the default value of 2.

If the RIJONX or RIJK approximation is used in the SCF, the same is also employed in the CP-SCF. Note, however, that the RI-K approximation is not efficient for these terms.

Note

When running RPA or the CP-SCF with COSX, the algorithm from ORCA-6.0.0 was hard-coded to a memory-hungry one, which was not the original intention. In ORCA-6.0.1 that was fixed, and the automatic one is now the default. This might cause very small differences to these results, specially if the number of densities is small. As a result, calculations give slightly different results with ORCA-6.0.0. This does not affect any "normal" energy, gradient or etc. calculations. Only those particular cases of RPA and CP-SCF.

2.23 Numerical Gradients

If the analytic gradient is not available, the numerical gradient can simply be requested via:

```
! NumGrad
```

as in the following example:

```
!CCSD(T) TZVPP
!Opt NumGrad
* int 0 1
C 0 0 0  0   0   0
O 1 0 0  1.2 0   0
H 1 2 0  1.1 120 0
H 1 2 3  1.1 120 180
*
```

NOTE

- Be aware that the numerical gradient is quite expensive. The time for one gradient calculation is equal to 6 × (number of atoms) × (time for one single point calculation).
- The numerical gradient can be calculated in a multi-process run, using a maximum of three times the number of atoms (see section *Parallel and Multi-Process Modules*).

More details on various options, geometry convergence criteria and the like are found in section *Geometry Optimizations*.

2.23.1 Some Notes and Tricks

Note

- `TightSCF` in the SCF part is set as default to avoid the buildup of too much numerical noise in the gradients.
- Even if the optimization does not converge, the ORCA output may still end with "****ORCA TERMINATED NORMALLY****". Therefore do not rely on the presence of this line as an indicator of whether the geometry optimization is converged! Rather, one should instead rely on the fact that, an optimization job that terminates because the maximum number of iterations has been reached, will generate the following output message:

Warning

```
The optimization did not converge but reached the maximum number of
optimization cycles. Please check your results very carefully.
```

While a successfully converged job will generate the following message instead:

```
************************HURRAY********************
***         THE OPTIMIZATION HAS CONVERGED      ***
*******************************************************
```

Tip

- In rare cases the redundant internal coordinate optimization fails. In this case, you may try to use `COPT` (optimization in Cartesian coordinates). This will likely take many more steps to converge but should be stable.
- For optimizations in Cartesian coordinates the initial guess Hessian is constructed in internal coordinates and thus these optimizations should converge only slightly slower than those in internal coordinates. Nevertheless, if you observe a slow convergence behaviour, it may be a good idea to compute a Hessian initially (perhaps at a lower level of theory) and use `InHess read` in order to improve convergence.
- At the beginning of a TS optimization more information on the curvature of the PES is needed than a model Hessian can give. The best choice is analytic Hessian, available for HF, DFT and MP2. In other cases (e.g. CAS-SCF), the numerical evaluation is necessary. Nevertheless you do not need to calculate the full Hessian when starting such a calculation. With ORCA we have good experience with approximations to the exact Hessian. Here it is recommended to either directly combine the TS optimization with the results of a relaxed surface scan or to use the Hybrid Hessian as the initial Hessian, depending on the nature of the TS mode. Note that these approximate Hessians do never replace the exact Hessian at the end of the optimization, which is always needed to verify the minimum or first order saddle point nature of the obtained structure.

2.23.2 Numerical Gradients

If you want to use numerical instead of analytic gradients you have to use

```
! NumGrad
```

in your input file. Additionally the settings for the numerical differentiation can be changed:

```
%numgrad
  CentralDiff true # You should use two-sided numerical differentiation, but it
                   # is possible to switch to one-sided numerical differentiation.
  DX 0.005         # Increment in Bohr for the differentiation.
  TransInvar  true # Take advantage of translation invariance
 end
```

CHAPTER

THREE

MODEL CHEMISTRIES

3.1 Wavefunction Types: RHF/RKS, UHF/UKS, ROKS and more

The basic variable that controls the type of wavefunction to be computed is the variable `HFTyp` in the `%scf` block. If nothing is specified for `HFTyp`, the program will check the multiplicity given in the input: for closed-shell molecules with multiplicity 1, RHF/RKS is assumed; for open shell molecules with multiplicity larger than 1, UHF/UKS is invoked. `RHF` will lead to a spin restricted closed-shell type computation [180] whereas UHF does a spin unrestricted calculation.

For DFT calculations, `RKS`, `UKS` and `ROKS` can be used as synonyms for `RHF`, `UHF` and `ROHF`. The restricted open-shell DFT method (`ROKS`) is only operative for high-spin states that have n unpaired electrons and $S = n/2$. `UKS` wavefunctions will not be spin-purified.

```
%scf
  HFTyp  RHF     # closed-shell (RKS for DFT)
         UHF     # unrestricted open-shell (UKS for DFT)
         ROHF    # restricted open-shell (ROKS for DFT)
         CASSCF  # complete active space SCF
end
```

In certain cases you may want to run open-shell systems with RHF/RKS to get a "half-electron" type wavefunction [181]. The total energy is not corrected! Sometimes these half-electron computations lead to acceptable convergence, and the resulting orbitals may be used as input for ROHF, UHF or MRCI calculations. Special attention should be taken for transition metal complexes where the RHF/RKS orbitals are quite different from ROHF or UHF orbitals, so that it is not recommended to over-interpret the wavefunctions from such calculations. The calculation is set up in the following way:

```
%method AllowRHF true end
# or simply: ! AllowRHF
```

3.1.1 ROHF Options

For ROHF calculations[6, 7, 182, 183, 184, 185, 186, 187, 188] the program will try to figure out what type of open-shell situation is present on the basis of the initial guess orbitals and their energies. Most "simple" cases are well recognized, but sometimes a little help from the user is needed.

The simplest ROHF case is the `HIGHSPIN` case, where all unpaired electrons in the open-shell are ferromagnetically coupled (parallel to each other), resulting in the highest multiplicity possible. The user can request this case as follow:

```
%scf
  HFTyp       ROHF
  ROHF_case   HIGHSPIN
  ROHF_NEl[1] 4          # number of electrons in the open-shell
end
```

The ROHF code also has a very powerful feature that goes back to insights of Mike Zerner [189, 190]. It can average over either *all* states of a given configuration (`CAHF`) or *all states of a given spin* for a given configuration (`SAHF`).

Especially, the SAHF feature gives you easy access to most degenerate high symmetry situations and the orbitals resulting from such calculations will be very convenient as input for CI calculations.

```
%scf
  HFTyp          ROHF
  ROHF_case      CAHF     # configuration averaged HF
                 SAHF     # spin averaged HF
  ROHF_NumOp     3        # number of operators (3, 2 or 1)
  ROHF_NOrb[1]   2,1      # number of orbitals in each open-shell
  ROHF_NEl[1]    1,1      # number of electrons in each open-shell
end
```

The hypothetical example below could represent an excited state of an octahedral d^3 transition metal complex. In this case there are five open-shell orbitals. The first three open-shell orbitals contain two electrons and the last two one electron. The input for a SAHF calculation is identical, just replace CAHF with SAHF.

```
%scf
  HFTyp          ROHF
  ROHF_case      CAHF    # configuration averaged HF
  ROHF_NumOp     3       # 3 operators in this case: closed, open1, open2
  ROHF_NOrb[1]   3,2     # 3 orbitals in first open shell, 2 in the second
  ROHF_NEl[1]    2,1     # 2 electrons in first open shell, 1 in the second
end
```

Another feature of the ROHF code is the ability to converge the SCF to a given Configuration State Function (CSF-ROHF) [191]. In this way one can approach results from MCSCF calculations. This can be requested in two ways.

The user can give a specific coupling situation.

```
%scf
  HFTyp       ROHF
  ROHF_CASE   USER_CSF           # User defined CSF
  ROHF_REF    {1 1 -1 -1} end    # CSF to be converged to a 2 open shell␣
↪antiferromagnetically coupled
end
```

Or the user can give how many orbitals per shell. Where each open-shell will couple with antiparallel spin with the previous one.

```
%scf
  HFTyp        ROHF
  ROHF_CASE    AF_CSF           # User defined CSF
  ROHF_AFORBS  2,2              # Coupling Situation
end
```

As an example, one can think of a Fe(III) dimer, where each center is locally high spin, but they couple antiferromagnetically to each other. In order to get the ROHF solution for this system, first one needs a set of initial guess orbitals. The guess orbitals can be obtained either from the QROs of an UHF calculation, or a high spin ROHF calculation, or even a SAHF or CAHF. Independently of the method used, the orbitals need to be localized and ordered in a way that the set of 3d orbitals of each iron are grouped together in sequence. From this, one can run a CSF-ROHF calculation for the antiferromagnetic CSF as shown bellow:

```
%scf
  HFTyp       ROHF
  ROHF_CASE   USER_CSF                          # User defined CSF
  ROHF_REF    {1 1 1 1 1 -1 -1 -1 -1 -1} end    # 2 open shells coupling antiparallel
end

or

%scf
```

(continues on next page)

(continued from previous page)

```
  HFTyp       ROHF
  ROHF_CASE   AF_CSF                          # User defined CSF
  ROHF_AFORBS 5,5                             # 2 open shells coupling antiparallel
end
```

The CSF-ROHF procedure can recognize doubly occupied and virtual orbitals in the definition of the CSF when the `USER_CSF` case is invoked. When detected, these orbitals will be rotated out of the open-shell defined in the ROHF method and the calculation will run normally:

```
%scf
  HFTyp      ROHF
  ROHF_CASE  USER_CSF                    # User defined CSF
  ROHF_REF   {1 1 2 1  -1 -1 0 -1} end   # the DOMO will be rotated to the closed-
↪shell and
                                         # the VMO will be rotated to the virtual␣
↪space.
end
```

The user can also directly input the ROHF variables by means of the `ROHFOPT Case User` keyword. For example, for the high spin case with three electrons in three orbitals, it gives two operators with vector coupling coefficients $a = 1$ and $b = 2$ (Zerner convention).

```
%scf
  HFTyp      ROHF
  ROHFOP  Case    User  # manual input of ROHF variables
    Nop     2           # number of operators
    Norb[1] 3           # number of open-shell orbitals
    Nel[1]  3           # number of open-shell electrons
    A[1,1]  1           # Coulomb vector in the open shell
    B[1,1]  2           # Exchange vector in the open shell
  end
end
```

One awkward feature of the ROHF theory is that the Fock operator is somewhat arbitrarily defined. Different choices lead to the same wavefunction, but have different convergence properties that may vary from system to system. ORCA thus lets the user choose the desired variant. Playing around with these choices may turn a divergent or slowly converging ROHF calculation into a successful calculation!

The `ROHF_Restrict` feature is another feature that may be useful. If you suspect that the ROHF calculation does not converge because an open-shell and a closed-shell orbital are flipping back and forth, you can try to avoid this behavior by choosing `ROHF_Restrict true`. Of course there is no guarantee that it will work, and no guarantee that the system stays in the desired state. However, it decreases the chances of large, uncontrolled steps.

```
%scf
  ROHF_Mode 0   # construct F according to Pulay (default)
            1   # construct F as in the Gamess program
            2   # construct F according to Kollmar

  ROHF_Restrict false # restrict orbital interchanges and off-diagonal elements
                      # (default=false)

  # a complete list of ROHF variables
  ROHFOP
    Case      User   # manual input of ROHF variables
    Nop       2      # number of operators
    Norb[1]   3      # number of open-shell orbitals
    Nel[1]    3      # number of open-shell electrons
    A[1,1]    1      # Coulomb vector in the open shell
    B[1,1]    2      # Exchange vector in the open shell
    Mode      2      # use the Kollmar operator
    Restrict  false  # do not restrict
```

(continues on next page)

(continued from previous page)

```
  end
end
```

3.1.2 UHF Natural Orbitals

The program can produce the UHF natural orbitals (UNOs). With these, the open-shell wavefunction can be pictured conveniently. The syntax is simple:

```
%scf
  UHFNO true
end
# or simply: ! UNO
```

There are various printing options for UNOs described in the output section *Population Analysis*. The UNOs can also be plotted as described in the plots section *Orbital and Density Plots*. In general the program stores a file `BaseName.uno`, where `BaseName` is by default the name of your input file with `.inp` stripped off. Accordingly, the gbw file is named `BaseName.gbw`. The `.uno` file is a normal gbw file that contains the geometry, basis set and the UNO orbitals. It could be used, for example, to start a ROHF calculation.

3.1.3 UNO, UCO and QROs input

For calculations on open-shell systems we recommend to use the keywords `!UNO !UCO` in the input line. This will generate quasi-restricted molecular orbitals `QRO`, unrestricted natural spin-orbitals `UNSO`, unrestricted natural orbitals `UNO` and unrestricted corresponding orbitals `UCO`. Moreover, it will print the UCO overlaps in the output, which can provide very clear information about the spin-coupling in the system. Below an example of the input and section of the output is provided.

```
!B3LYP def2-SVP UNO UCO TightSCF
```

The UCO overlap section in the output will look like:

```
***UHF Corresponding Orbitals were saved in MyJob.uco***

----------------------
Orbital    Overlap(*)
----------------------
.
.
.
96:       0.99968
97:       0.99955
98:       0.99947
99:       0.99910
100:      0.99873
101:      0.99563
102:      0.74329
103:      0.00000
```

The overlap that corresponds to a value usually less than 0.85 denotes a spin-coupled pair. Whereas, values close to 1.00 and 0.00 refers to the doubly occupied and singly occupied orbitals respectively.

3.2 Hartree Fock Theory

The Hartree-Fock (HF) method[192, 193, 194] is a basic *ab-initio* quantum mechanical method that is used to approximate many-body electronic wavefunctions within mean field theory. Many post-Hartree-Fock methods like Coupled-Cluster typically make use of HF reference wavefunctions.

A basic Hartree-Fock calculation can be invoked via the simple input keyword `!HF`

```
! HF
```

or via the `%method` block.

```
%method
   method HF    # equivalent to method HFGTO
end
```

By default, ORCA will make use of the `def2-SVP` *basis set* if not specified else. Hartree-Fock calculations can be accelerated drastically by using different *RI approximations*.

See also

ORCA provides several choices of wavefunction types like resticted (RHF), unresticted (UHF), or unrestricted open-shell HF (ROKS). These options are documented in Section 3.1. HF is also an integral part of *hybrid-* and *double-hybrid* DFT.

Tip

Grimme and co-workers developed a low-cost method based on HF called *HF-3c* that can be employed for very large systems. Some HF-based *semi-empirical methods* like AM1 or PM3 are also available.

3.3 Density Functional Theory (DFT)

Due to its good cost-accuracy ratio, Kohn-Sham density functional theory (DFT) has become the workhorse of quantum chemistry. Over the last few decades, a whole zoo of density functionals has been developed that differ in terms of their construction and design philosophy. The so-called Jacob's Ladder picture (Fig. 3.1) coined by Perdew [195] categorizes density functionals according to their basic physical ingredients into rungs . With higher rungs. the cost of the given functionals increased but also higher accuracy is conceptionally expected. However, the accuracy of density functionals strongly varies and careful study of benchmark studies is generally recommended.

Fig. 3.1: Jacob's ladder picture of density functionals.

ORCA features various density functionals of each rung either implemented natively or via the *LibXC Functional Library*. These functionals can be combined with various *basis sets* and *relativistic approaches*. Further, ORCA features a useful toolkit of *resolution-of-identity (RI)* methods to speed-up DFT calculations.

For dispersion corrections that can be used with DFT and the respective available functional-specific parameters refer to the *Dispersion Corrections section*.

Also note that some functionals like the SCAN and Minnesota families may require enhanced *numerical integration grid settings*.

3.3.1 Basic Usage

All implemented DFT methods can be invoked via their respective keywords, e.g. `!PBE` using the simple input line:

```
! PBE
```

Warning

If only a functional is defined, various default settings will be used including a small def2-SVP basis set, an appropriate *RI* treatment and default *numerical integration grids*! To achieve higher accuracy, specifically the *basis set* should be adjusted!

Many functionals can also be invoked via the `%method` block even though double-hybrid functionals being not available due to technical reasons:

```
%method
   Method DFT        # equivalent to method DFGTO
   Functional PBE
end
```

ORCA further provides some more utility to construct and modify functionals via the `%method` block. These expert options are described in the respective sections of each functional rung. Various other functionals can also be invoked via the *LibXC Functional Library*.

3.3.2 Local Functionals

Local exchange-correlation functionals are the most basic examples of functionals constructed according to equation (3.1):

$$E_{\mathrm{XC}} = E^{\mathrm{X}}_{\mathrm{LSD}} + E^{\mathrm{C}}_{\mathrm{LSD}} \tag{3.1}$$

They typically the original Slater exchange functional with different local correlations functionals like VWN-V[196] or PW-LDA[197]. All available local functional that are available in ORCA can be found in Table 3.1.

Table 3.1: List of available local functionals.

Functional	Keywords	Description
HFS	`HFS`	Hartree–Fock–Slater Exchange only functional
LDA or LSD	`LDA`, `LSD`	Local density approximation (defaults to VWN5)
VWN-V [196]	`VWN`,`VWN5`	Vosko-Wilk-Nusair local density approx. parameter set "V"
VWN-III [196]	`VWN3`	Vosko-Wilk-Nusair local density approx. parameter set "III"
PW-LDA [197]	`PWLDA`	Perdew-Wang parameterization of LDA

3.3.3 Gradient Corrected Functionals: (meta-)GGAs

The next rung corresponds to gradient corrected functional of the generalized gradient approximation (GGA) and the meta-generalized gradient approximation (meta-GGA) types. A simplified exchange-correlation energy expression for these types of functionals is given in (3.2). (meta-)GGAs are specifically useful for *geometry optimizations* and screening purposes due to their comparably small computational cost.

$$E_{\mathrm{XC}} = E^{\mathrm{X}}_{\mathrm{LSD}} + E^{\mathrm{X}}_{\mathrm{GGA}} + E^{\mathrm{C}}_{\mathrm{LSD}} + E^{\mathrm{C}}_{\mathrm{GGA}} \tag{3.2}$$

Table 3.2: List of available (meta-)GGA functionals.

Functional	Keyword	Description
GGAs		
BP86 [198, 199]	BP86	Becke '88 exchange and Perdew '86 correlation
BLYP [200]	BLYP	Becke '88 exchange and Lee-Yang-Parr correlation
OLYP [201]	OLYP	Handy's "optimal" exchange and Lee-Yang-Parr correlation
GLYP [202]	GLYP	Gill's '96 exchange and Lee-Yang-Parr correlation
XLYP [203]	XLYP	The Xu and Goddard exchange and Lee-Yang-Parr correlation
PW91 [204]	PW91	Perdew-Wang '91 GGA functional
mPWPW [205]	MPWPW	Modified PW exchange and PW correlation
mPWLYP [205]	MPWLYP	Modified PW exchange and LYP correlation
PBE [206]	PBE	Perdew-Burke-Erzerhoff GGA functional
RPBE [207]	RPBE	"Modified" PBE
revPBE [208]	REVPBE	"Revised" PBE
RPW86PBE [209]	RPW86PBE	PBE correlation with refitted Perdew '86 exchange
PWP [204]	PWP	Perdew-Wang '91 exchange and Perdew '86 correlation
composite		
B97-3c [210]	B97-3C	Composite DFT method by Grimme et al.
meta-GGAs		
B97M-V [211]	B97M-V	Head-Gordon's DF B97M-V with VV10 nonlocal correlation
B97M-D3(BJ) [212]	B97M-D3BJ	Modified version of B97M-V with D3BJ correction by Najibi and Goerigk
B97M-D4 [213]	B97M-D4	Modified version of B97M-V with DFT-D4 correction by Najibi and Goerigk
SCAN [214]	SCANFUNC	Perdew's SCAN functional
rSCAN [215]	RSCAN	Regularized SCAN functional
r^2SCAN [216]	R2SCAN	Regularized and restored SCAN functional by Furness, Sun et al.
M06-L [217]	M06L	The Minnesota M06-L meta-GGA functional
TPSS [218]	TPSS	The TPSS meta-GGA functional
revTPSS [219, 220]	REVTPSS	Revised TPSS meta-GGA functional
composite		
r^2SCAN-3c [221]	R2SCAN-3C	Composite DFT method by Grimme et al.

Customization of Basic Functionals

Basic exchange-correlation functionals can also be constructed from their components via the `%method` block using the `Exchange`, `Correlation`, and `LDAOpt` keywords. These controle the exchange and correlation parts of the functional as well as the local part of the correlation functional, respectively. For example BP86 (`!BP86`) is constructed from Becke's '88 exchange via `X_BECKE` and Perdew's '86 correlation functional via `C_P86`. In ORCA, it further employs Perdew and Wang's LDA reparameterization[197] `PWLDA` for the local part of the correlation functional via `C_PWLDA`:

```
%method
   Method DFT
   Exchange     X_BECKE
   Correlation  C_P86
   LDAOpt       C_PWLDA
end
```

For a full list of available components see Table 3.3.

The correct choice of the local part of the correlation functional is crucial to ensure reproducibility of the results. In ORCA, almost all functionals choose `PWLDA`[197] as the underlying LDA functional as it represents a slightly better fit to the uniform electron gas than VWN. For VWN (Vosko, Wilk, and Nusair[196]), several parameterizations exist.

In their classic paper give two sets of parameters – one in the main body (parametrization of RPA results; known as VWN-III) and one in their table 5 of correlation energies (parametrization of the Ceperley/Alder Monte Carlo results; known as VWN-V). The results from PW-LDA are very similar to those of VWN-V, whereas VWN-III yields quite different values. Over the years, the usage of different parameterizations for the same functional name throughout different quantum chemistry programs caused some confusion and deviating results. A prominent example is the different implementation of the B3LYP *hybrid functional* within ORCA and Gaussian. While ORCA uses VNW-V consistent with the TURBOMOLE program[222, 223, 224], the Gaussian program[225] employs VWN-III. Using the `LDAOpt` keyword, both variants can be constructed:

```
# consistent with TURBOMOLE
%method
   Method DFT
   Functional B3LYP
   LDAOpt C_VWN5
end
```

```
# consistent with Gaussian
%method
   Method DFT
   Functional B3LYP
   LDAOpt C_VWN3
end
```

For convenience both functionals can be invoked separately via the simple input keywords `!B3LYP` for the ORCA/TURBOMOLE variant and `!B3LYP/G` for the Gaussian variant.

Note

A special situation arises if LYP is the correlation functional [226] since LYP is *not* a correction to the correlation, but rather includes the full correlation. It is therefore used in the B3LYP functional as:

$$E^{\mathrm{C}}_{\mathrm{B3LYP}} = E^{\mathrm{C}}_{\mathrm{LSD}} + c\left(E^{\mathrm{C}}_{\mathrm{LYP}} - E^{\mathrm{C}}_{\mathrm{LSD}}\right) \tag{3.3}$$

Some functionals further incorporate empirical parameters that can be changed to improve agreement with experiment. Currently, there are a few parameters that can be changed (other than the parameters used in the *hybrid functionals*, which are described later).

The first of these parameters is α of Slater's X_α method. Theoretically, it has a value of 2/3 and this is used in the HFS and LSD functionals. However, the exchange contribution is underestimated by about 10% by this approximation (quite significant!) and a value around 0.70-0.75 is recommended for molecules.

The second parameter is the β for Becke's gradient-corrected exchange functional. Becke determined the value 0.0042 by fitting the exchange energies for rare gas atoms. There is some evidence that with smaller basis sets, a slightly smaller value such as 0.0039 gives improved results for molecules.

The next parameter is the value κ, which occurs in the PBE exchange functional. It has been given the value 0.804 by Perdew et al. in order to satisfy the Lieb-Oxford bound. Since then, other workers have argued that a larger value for this parameter (around 1.2) gives better energetics, which is explored in the revPBE functional. Note, however, that while revPBE gives slightly better energetics, it also gives slightly poorer geometries.

The last two parameters are also related to PBE. Within the PBE correlation functional, there is β_C (not to be confused with the β exchange parameter in Becke's exchange functional), whose original value is $\beta_C = 0.066725$. Modified variants exist with different β_C values, e.g., the PBEsol functional and the *PBEh-3c composite method*. Furthermore, the μ parameter in the PBE exchange functional may be modified. In the original formulation, it is related to β_C via $\mu = \beta_C \frac{\pi^2}{3}$, but has been modified in later variants as well.

Table 3.3: `%method` block input keywords to modify basic exchange-correlation functionals.

Keyword	Options	Description
`Exchange`	`X_NOX`	No exchange
	`X_SLATER`	Slater's local exchange
	`X_BECKE`	Becke's '88 exchange
	`X_wB88`	Short-range Becke '88 exchange for range-separated functionals
	`X_G96`	Gill '96 gradient exchange
	`X_PW91`	Perdew-Wang (PW) '91 gradient exchange
	`X_mPW`	Adamo-Barone modification of PW exchange
	`X_PBE`	Perdew-Burke-Ernzerhof (PBE) exchange
	`X_RPBE`	Revised PBE exchange
	`X_OPTX`	Hoe/Cohen/Handy's optimized exchange
	`X_X`	Xu/Goddard exchange
	`X_TPSS`	TPSS meta-GGA exchange
	`X_B97D`	Grimme's modified exchange for the B97-D GGA
	`X_B97BECKE`	Becke's original exchange for the B97 hybrid
	`X_SCAN`	Perdew's constrained exchange for the SCAN meta-GGA
	`X_RSCAN`	Perdew's constrained exchange for the rSCAN meta-GGA
	`X_R2SCAN`	Perdew's constrained exchange for the r^2SCAN meta-GGA
`Correlation`	`C_NOC`	No correlation
	`C_VWN5`	Local Vosko-Wilk-Nusair correlation (parameter set "V")
	`C_VWN3`	Local Vosko-Wilk-Nusair correlation (parameter set "III")
	`C_PWLDA`	Local PW correlation
	`C_P86`	Perdew '86 correlation
	`C_PW91`	PW '91 correlation
	`C_PBE`	PBE correlation
	`C_LYP`	LYP correlation
	`C_TPSS`	TPSS meta-GGA correlation
	`C_B97D`	Grimme's modified correlation for the B97-D GGA
	`C_B97BECKE`	Becke's original correlation for the B97 hybrid
	`C_SCAN`	Perdew's constrained correlation for the SCAN meta-GGA
	`C_RSCAN`	Perdew's constrained correlation for the rSCAN meta-GGA
	`C_R2SCAN`	Perdew's constrained correlation for the r^2SCAN meta-GGA
`LDAOpt`	`C_NOC`	Sets the local part of any correlation functional to no correlation
	`C_PWLDA`	Sets the local part of any correlation functional to PWLDA
	`C_VWN5`	Sets the local part of any correlation functional to VWN5
	`C_VWN3`	Sets the local part of any correlation functional to VWN3
Specific Parameters		
`XAlpha`	`<real>`	Modifies Slater's α parameter (default 2/3)
`XBeta`	`<real>`	Modifies Becke's β parameter (default 0.0042)
`XKappa`	`<real>`	Modifies the PBE(exchange) κ parameter (default 0.804)
`XMuePBE`	`<real>`	Modifies the PBE(exchange) κ parameter (default 0.21952)
`CBetaPBE`	`<real>`	Modifies the PBE(correlation) beta parameter (default 0.066725)

3.3.4 Hybrid DFT

The idea of replacing a portion of the DFT exchange by exact exchange from Hartree-Fock theory (HFX) was introduced by Becke in 1993[227, 228]. This scheme is typically referred to as hybrid DFT and the most simple formulation of a one-parameter hybrid functional is depicted in equation (3.4)

$$E_{\text{hyb}}^{\text{XC}} = aE_{\text{HF}}^{\text{X}} + (1-a)E_{\text{GGA}}^{\text{X}} + E_{\text{GGA}}^{\text{C}} \tag{3.4}$$

A more general description is given by the adiabatic connection model (ACM). In the ACM, hybrid functionals are constructed with three parameters according to equation (3.5):

$$E_{\text{XC}} = aE_{\text{HF}}^{\text{X}} + (1-a)\,E_{\text{LSD}}^{\text{X}} + bE_{\text{GGA}}^{\text{X}} + E_{\text{LSD}}^{\text{C}} + cE_{\text{GGA}}^{\text{C}} \tag{3.5}$$

Here, E_{XC} is the total exchange/correlation energy, $E^{\mathrm{X}}_{\mathrm{HF}}$ is the Hartree-Fock exchange, $E^{\mathrm{X}}_{\mathrm{LSD}}$ is the local (Slater) exchange, $E^{\mathrm{X}}_{\mathrm{GGA}}$ is the gradient correction to the exchange, $E^{\mathrm{C}}_{\mathrm{LSD}}$ is the local, spin-density based part of the correlation energy, and $E^{\mathrm{C}}_{\mathrm{GGA}}$ is the gradient correction to the correlation energy.

Over the years, various strategies to construct hybrid functionals have been developed. These range from physics motivated approaches relying on as few as possible parameters like the overall well-balanced PBE0 functional[229] to more empirical variants like the prominent three-parameter hybrid B3LYP[230]. Furthermore, various treatments of the HF exchange admixture were developed. The basic hybrids that use a static amount of HFX are typically referred to as *global hybrids* while functionals with variable amounts of HFX include *range-separated hybrids* and local hybrids. Various *global* and *range-separated hybrid functionals* are available in ORCA. Local hybrids may be implemented in a future version.

ORCA also provides a sophisticated infrastructure to create and *customize global* and *range-separated hybrid functionals*.

Note

As by theory hybrid functionals are significantly more costly than (meta-)GGAs, ORCA uses the sophisticated *RIJCOSX* algorithm as a default to speed up hybrid DFT calculations.

Global Hybrid Functionals

Global hybrid functionals employ a fixed amount of Hartree-Fock exact exchange. ORCA features various representatives with different compositions and amounts of HFX. The optimal amount of HFX is controversially discussed in the literature and many empirical functionals were constructed by fitting it to reproduce an experimental property. However, it has been argued from a theoretical viewpoint that the optimal mixing of HF exchange is 25% [231] and large amounts of HFX are known to be disadvantageous for transition metal complexes.

A list of natively implemented global hybrid functionals is given in Table 3.4.

Table 3.4: List of available global hybrid functionals.

Functional	Key-word	%HF	Description
hybrid GGAs			
B1LYP [232]	B1LYP	25	The one-parameter hybrid functional with Becke '88 exchange and Lee-Yang-Parr correlation (25% HF exchange)
B3LYP [230]	B3LYP	20	The popular B3LYP functional (20% HF exchange) as defined in the Turbo-Mole program system (LDAOPT C_VWN5)
B3LYP/G [230]	B3LYP/G	20	The popular B3LYP functional (20% HF exchange) as defined in the Gaussian program system (LDAOPT C_VWN3)
O3LYP [233]	O3LYP	11.61	The Handy hybrid functional
X3LYP [203]	X3LYP	21.8	The Xu and Goddard hybrid functional
B1P86 [199]	B1P86	25	The one-parameter hybrid version of BP86
B3P86 [227]	B3P86	20	The three-parameter hybrid version of BP86
B3PW91 [227]	B3PW91	20	The three-parameter hybrid version of PW91
PW1PW [229]	PW1PW	25	One-parameter hybrid version of PW91
mPW1PW [205]	MPW1PW	25	One-parameter hybrid version of mPWPW
mPW1LYP [205]	MPW1LYP	25	One-parameter hybrid version of mPWLYP
PBE0 [229]	PBE0	25	One-parameter hybrid version of PBE
revPBE0 [234]	REVPBE0	25	"Revised" PBE0
revPBE38 [234]	REVPBE38	37.5	"Revised" PBE0 with 37.5% HF exchange
BHandHLYP [227]	BHANDHLYP	50	Half-and-half hybrid functional by Becke
hybrid meta-GGAs			
M06 [235]	M06	27	
M06-2X [235]	M062X	54	
PW6B95 [236]	PW6B95	28	
TPSSh [218]	TPSSH	10	The hybrid version of TPSS (10% HF exchange)
TPSS0 [237]	TPSS0	25	A 25% exchange version of TPSSh that yields improved energetics
r^2SCANh [238]	R2SCANH	10	Global hybrid variant of r^2SCAN with 10% HF exchange
r^2SCAN0 [238]	R2SCAN0	25	Global hybrid variant of r^2SCAN with 25% HF exchange
r^2SCAN50 [238]	R2SCAN50	50	Global hybrid variant of r^2SCAN with 50% HF exchange
composite			
PBEh-3c [239]	PBEH-3C	42	Composite DFT approach by Grimme et al.
B3LYP-3c [240]	B3LYP-3C	20	Composite DFT approach by Grimme et al.

Customization of Global Hybrids

Global hybrid functional parameters can be customized via the `%method` block. For the customization of the basic exchange and correlation components see the *Customization of Basic Functionals* section. The important parameters for hybrid functionals are those used in the ACM model (cf. equation (3.5)). In this case, we will recreate the PBE0 functional (25% HFX) by using the ACM syntax:

```
! RIJCOSX
# PBE0
%method
   Method DFT
   Functional PBE
   ACM_A        0.25      # fraction of Hartree-Fock exchange
   ACM_B        0.75      # fraction of the GGA exchange
   ACM_C        1.00      # fraction of the GGA correlation
   ScalLDAC     1.00      # fraction of LDA correlation, must be equal to ACM_C
end
```

For simplicity, the ACM parameters can also be given as one string.

```
! RIJCOSX
# PBE0
%method
   Method DFT
   Functional PBE
   ACM       0.25, 1.00, 1.00   # as ACM   ACM-A, ACM-B, ACM-C
   ScalLDAC  1.00               # fraction of LDA correlation, must be equal to␣
↪ACM_C
end
```

As the ACM formalism can be non-intutive, ORCA provides an equivalent syntax to control the respective parameters.

```
! RIJCOSX
# PBE0
%method
   Method DFT
   Functional PBE
   ScalHFX        0.25      # fraction of Hartree-Fock exchange
   ScalDFX        0.75      # fraction of GGA exchange
   ScalGGAC       1.00      # fraction of GGA correlation
   ScalLDAC       1.00      # fraction of LDA correlation, must be equal to␣
↪ScalGGAC
end
```

Warning

Note that `Functional` chooses specific predefined values for the variables `Exchange`, `Correlation`, `LDAOpt` and `ACM` described below. If given as a simple input keyword, in some cases, it will also activate a dispersion correction. You can explicitly give these variables instead or in addition to `Functional`. However, make sure that you specify these variables *after* you have assigned a value to `Functional` or the others will be reset to the values chosen by `Functional`.

Table 3.5: `%method` block input keywords to modify global hybrid functionals.

Keyword	Options	Description
ACM	<real, real, real>	Defines the ACM-A, ACM-B, and ACM-C parameters
ACM_A	<real>	Defines the ACM-A parameter (fraction of Hartree-Fock exchange)
ACM_B	<real>	Defines the ACM-B parameter (fraction of GGA exchange)
ACM_C	<real>	Defines the ACM-C parameter (fraction of GGA correlation)
ScalHFX	<real>	Controls the fraction of Hartree-Fock exchange, equiv. to ACM_A
ScalDFX	<real>	Controls the fraction of GGA exchange, equiv. to ACM_B
ScalGGAC	<real>	Controls the fraction of GGA correlation, equiv. to ACM_C

Range-Separated Hybrid Functionals

Unlike *global hybrids*, range-separated hybrid (RSH) split the two-electron operator used for the exchange into a short-range (SR) and a long-range (LR) regime wherein the short-range part is typically described by DFT exchange and the long-range part by Hartree-Fock exchange. In ORCA, all range-separated hybrids use the error function splitting approach by Hirao and coworkers to distinguish both regimes (cf. equation (3.6)).[241] This allows the definition of DFT functionals that dominate the short-range part by an adapted exchange functional of *LDA*, *GGA or meta-GGA* level and the long-range part by Hartree-Fock exchange.

$$r_{12}^{-1} = \underbrace{\mathrm{erfc}(\mu \cdot r_{12}) \cdot r_{12}^{-1}}_{\mathrm{SR}} + \underbrace{\mathrm{erf}(\mu \cdot r_{12}) \cdot r_{12}^{-1}}_{\mathrm{LR}} \tag{3.6}$$

where $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$ and $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$. Note that the splitting is only applied to the exchange contribution; all other contributions (one-electron parts of the Hamiltonian, the electron-electron Coulomb interaction and the approximation for the DFT correlation) are not affected. Later, Handy and coworkers[242] generalized the ansatz to:

$$r_{12}^{-1} = \underbrace{\frac{1 - [\alpha + \beta \cdot \mathrm{erf}(\mu \cdot r_{12})]}{r_{12}}}_{\mathrm{SR}} + \underbrace{\frac{\alpha + \beta \cdot \mathrm{erf}(\mu \cdot r_{12})}{r_{12}}}_{\mathrm{LR}} \tag{3.7}$$

This form of the splitting used in ORCA is shown visually (according to Handy and coworkers) in Fig. 3.2.

Fig. 3.2: Graphical description of the range-separation ansatz. The gray area corresponds to Hartree-Fock exchange. α and β follow Handy's terminology [242].

Important

Currently, RSH functionals can be used for integral-direct single-point calculations, calculations involving the first nuclear gradient (i.e. geometry optimizations), frequency calculations, TDDFT, TDDFT nuclear gradient, and EPR/NMR calculations. As with the *global hybrids*, *RIJCOSX* is used by default to drastically speed up the calculations. Alternatively, only RIJONX can be used.

A list of available range-separated hybrid functionals is given in Table 3.6

Table 3.6: List of available range-separated hybrid functionals. μ is the range-separation parameter.

Functional	Keyword	%HFX	μ / bohr^{-1}	% fixed DFX
hybrid GGAs				
ωB97 [243]	WB97	0 – 100	0.40	—
ωB97X [243]	WB97X	15.7706 – 100	0.30	—
ωB97X-V [244]	WB97X-V	16.7 – 100	0.30	—
ωB97X-D3 [245]	WB97X-D3	19.5728 – 100	0.25	—
ωB97X-D3(BJ) [212]	WB97X-D3BJ	16.7 – 100	0.30	—
ωB97X-D4 [213]	WB97X-D4	16.7 – 100	0.30	—
ωB97X-D4rev[49]	WB97X-D4REV	16.7 – 100	0.30	—
CAM-B3LYP [242]	CAM-B3LYP	19 – 65	0.33	35%
LC-BLYP [246]	LC-BLYP	0 – 100	0.33	—
LC-PBE [241]	LC-PBE	0 – 100	0.47	—
hybrid meta-GGAs				
ωB97M-V [247]	WB97M-V	15 – 100	0.30	—
ωB97M-D3(BJ) [212]	WB97M-D3BJ	15 – 100	0.30	—
ωB97M-D4 [213]	WB97M-D4	15 – 100	0.30	—
ωB97M-D4rev [49]	WB97M-D4REV	15 – 100	0.30	—
ωr^{2}SCAN [248]	WR2SCAN	0 – 100	0.30	—
composite				
ωB97X-3c [49]	WB97X-3C	16.7 – 100	0.30	—

Customizing Range-Separated Hybrids

Range-separated hybrid functional parameters can also be customized via the `%method` block. For the customization of the basic exchange and correlation components see the *Customization of Basic Functionals* section. Basic hybrid-related keywords and the respective ACM model are described in the *Hybrid DFT* and *Customization of Global Hybrids* sections.

In this example, we will recreate the LC-PBE functional:

```
! RIJCOSX
# LC-PBE
%method
   Method DFT
   Exchange X_LRCPBE
   Correlation C_PBE
   LDAOpt C_PWLDA
   ACM_A        0.00   # fraction of static Hartree-Fock exchange
   ACM_B        0.00   # fraction of static GGA exchange
   ACM_C        1.00   # fraction of GGA correlation
   ScalLDAC     1.00   # fraction of LDA correlation, must be equal to ScalGGAC
   RangeSepEXX  true   # activates range-separated exchange
   RangeSepMu   0.47   # range-separation parameter, should be >0
   RangeSepScal 1.00   # fraction of variable HFX, should sum to 1 with ACM-A and␣
↪ACM-B
end
```

Which is equivalent to the `ScalX` syntax described before.

```
! RIJCOSX
# LC-PBE
%method
   Method DFT
   Exchange X_LRCPBE
   Correlation C_PBE
```

(continues on next page)

(continued from previous page)

```
   LDAOpt C_PWLDA
   ScalHFX      0.00   # fraction of static Hartree-Fock exchange
   ScalDFX      0.00   # fraction of static GGA exchange
   ScalGGAC     1.00   # fraction of GGA correlation
   ScalLDAC     1.00   # fraction of LDA correlation, must be equal to ScalGGAC
   RangeSepEXX  true   # activates range-separated exchange
   RangeSepMu   0.47   # range-separation parameter, should be >0
   RangeSepScal 1.00   # fraction of variable HFX, should sum to 1 with ScalHFX␣
↪and ScalDFX
end
```

Warning

The amount of fixed DFT Exchange (`ACM_B` or `ScalDFX`) can only be changed for CAM-B3LYP and LC-BLYP. Accordingly, ACM-B is ignored by the ωB97 approaches as no corresponding μ-independent exchange functional has been defined. While it is technically possible to choose an exchange functional that has no μ-dependence, this makes no sense conceptually.

Table 3.7: `%method` block input keywords to modify range-separated hybrid functionals.

Keyword	Options	Description
`RangeSepEXX`	`true`	Activates range-separated exchange
`RangeSepMu`	`<real>`	Controls the range-separation parameter, should be >0
`RangeSepScal`	`<real>`	Controls the fraction of variable Hartree-Fock exchange

3.3.5 Double-Hybrid DFT

In addition to mixing the HF-exchange into a given density functional, Grimme has proposed to mix in a fraction of the *MP2 correlation energy* as calculated with hybrid DFT orbitals [249]. Such functionals may be referred to as "double-"hybrid functionals. The original expression by Grimme is:

$$E_{\mathrm{XC}} = aE_{\mathrm{X}}^{\mathrm{HF}} + (1-a)\,E_{\mathrm{X}}^{\mathrm{DFT}} + (1-c)\,E_{\mathrm{C}}^{\mathrm{DFT}} + cE_{\mathrm{C}}^{\mathrm{MP2}} \tag{3.8}$$

The first proposed double-hybrid functional by Grimme was B2PLYP.[249] This double-hybrid functional utilizes the B88 exchange functional and the LYP correlation functionals. It introduces 53% of Hartree-Fock exchange ($a = 0.53$) and replaces 27% of the DFT correlation energy by MP2 correlation energy ($c = 0.27$). Since then many other DH-DFT approaches that deviate from the original theme were developed. For example Grimme proposed DH functionals like PWPB95[250] that use spin-component-scaled MP2. Further, functionals whose composition was empirically fitted against databases without any physical constraints or restrictions as previously proposed in (3.8) like Martin's DSD (dispersion corrected, spin-component scaled double-hybrid) functionals[251, 252, 253] were developed and refined with revDSD-PBEP86-D4[254] being one of the most well known representatives. Functionals of the XYGn family[255] or Head-Gordon's ωB97M(2)[256] employ differently constructed densities for the DFT and the perturbation-theory parts with the latter also basing on a *range-separated hybrid functional*. In principle, other correlated methods like dRPA or higher order MP theory may be used within the double hybrid scheme. However, the most common approach employs classical *MP2* or its spin-component-scaled variants. The *DLPNO approximation to MP2* is also available within ORCA and can be used to massively speed up the energy and gradient calculations for large systems. The accuracy and efficiency of DLPNO-DHs was extensively studied by Bursch, Neese and co-workers[257]. Common to almost all double-hybrid functionals is a comparably large amount of Hartree-Fock exchange of 50% and more. Most classical double-hybrid functionals following the scheme by Grimme can be utilized for single point calculations, analytic gradients (and thus also geometry optimizations[258], relaxed scans, and transition state searches), dipole moments and other first order properties, magnetic second order properties (chemical shifts and g-tensors without the frozen core approximation in the MP2 part), as well as for numerical polarizabilities and frequencies.

Some double-hybrid functionals were further optimized for excited state applications and will make use of a *Doubles Corrections* to correct the singles-only excited states within the *time-dependent DFT (TD-DFT)* framework. Some of them, like SCS-PBE-QIDH only make use of spin-component-scaling for the CIS(D) component of the calculation. However, many of them have recently also been tested for ground-state properties.[259]

Note

ORCA employs the *RIJCOSX* and *RI-MP2* by default. If this is not wanted, RI can be deactivated generally by using the `!NORI` keyword or if only the RI approximation of the MP2 part should be deactivated via the `%mp2` block:

```
%mp2
   RI false
end
```

Indeed, there is hardly ever any reason to use conventional MP2.

Note

Note that per default no MP2 response density is calculated as its construction adds significant overhead. In this case, only the double-hybrid energy is correct. The computation of the MP2 response density that is consistent with first-order properties from analytic first-derivatives can be requested via the `%mp2 density relaxed end` statement in the `%mp2` block. The considerably less costly unrelaxed (expectation value-like) density may instead requested by `%mp2 density unrelaxed end`. However, this is not recommended since the changes to the relaxed density are considerable in our experience and the unrelaxed density has a much weaker theoretical status than its relaxed counterpart.

Global Double-Hybrid Functionals

Global double-hybrid functionals are based on the respective *global hybrid functionals* and employ static admixtures of *Hartree-Fock* exchange and *the MP2 correlation energy*. A list of currently implemented global double-hybrid functionals is given in Table 3.8. The revDSD-PBEP86-D4[254] by Martin and co-workers has proven to be accurate for most thermochemistry applications.

Warning

For some functionals of the DSD/DOD family, different parameters for the same functional name were provided by the authors. The employed parameterization is indicated by the `(year-of-publication)` suffix of the functional and the respective keywords. Also not that for some of them, the basic parameters change in presence of different *dispersion corrections*. Please carefully check, if the employed parameters agree with your expectations!

Hint

For most natively implemented double-hybrids you can simple add the `DLPNO-` prefix to the respective functional simple input keyword, e.g. `DLPNO-B2PLYP`. For all others, the DLPNO treatment can be activated via the `%mp2` block using the `DLPNO true` keyword.

Table 3.8: List of available global double-hybrid functionals. `PS` is the opposite-spin, `PT` the same-spin scaling parameter. Excited-states optimized functionals only employ spin-component-scaling for the CIS(D) doubles correction within the the TD-DFT framework. Same-spin indirect term (CTss), opposite-spin indirect term (CTos), same-spin direct term (CUss), and opposite-spin direct term (CUos) parameters are given in the footnotes.

Functional	Keywords	%HFX	%DFX	%GGAC	%MP2C	PS	PT
DSD-BLYP [251]	DSD-BLYP D3BJ	69	31	54	100	0.46	0.37
DSD-BLYP-NoDisp(201 [253]	DSD-BLYP, 2013	75	25	53	100	0.46	0.60
DSD-BLYP-D3(BJ)(2013 [253]	DSD-BLYP, 2013 D3BJ	71	29	54	100	0.47	0.40
DSD-PBEP86 [252]	DSD-PBEP8 D3BJ	70	30	43	100	0.53	0.25
DSD-PBEP86-NoDisp(201 [253]	DSD-PBEP8 2013	72	28	44	100	0.51	0.36
DSD-PBEP86-D3(BJ)(2013 [253]	DSD-PBEP8 2013 D3BJ	69	31	44	100	0.52	0.22
DSD-PBEB95-NoDisp [253]	DSD-PBEB9	70	30	52	100	0.48	0.22
DSD-PBEB95-D3(BJ) [253]	DSD-PBEB9 D3BJ	66	34	55	100	0.46	0.09
B2PLYP [249]	B2PLYP	53	47	73	27	–	–
mPW2PLYP [260]	mPW2PLYP	55	45	75	25	–	–
B2GP-PLYP [261]	B2GP-PLYP	65	35	64	36	–	–
B2K-PLYP [262]	B2K-PLYP	72	28	58	42	–	–
B2T-PLYP [262]	B2T-PLYP	60	40	69	31	–	–
B2NC-PLYP [263]	B2NC-PLYP	81	19	45	55	–	–
PWPB95 [250]	PWPB95	50	50	73.1	26.9	1.0	0.0

continues on next page

Table 3.8 – continued from previous page

Functional	Keywords	%HFX	%DFX	%GGAC	%MP2C	PS	PT
PBE-QIDH [264]	PBE-QIDH	69	31	67	33	–	–
PBE0-DH [265]	PBE0-DH	50	50	87.5	12.5	–	–
revDSD-PBEP86(202 [254]	REVDSD-PI 2021	69	31	42.24	100	0.5935	0.0566
revDSD-PBEP86-D4(2021) [254]	REVDSD-PI 2021	69	31	42.24	100	0.5935	0.0566
revDOD-PBEP86(202 [254]	REVDOD-PI 2021	69	31	43.01	100	0.6131	0.0
revDOD-PBEP86-D4(2021) [254]	REVDOD-PI 2021	69	31	43.01	100	0.6131	0.0
SOS1-r^{2}SCAN-CIDH [248]	R2SCAN-C:	55	45	83.33	16.67	1.333	0.0
SOS1-r^{2}SCAN-QIDH [248]	R2SCAN-Q:	69.34	30.76	66.67	33.33	1.333	0.0
SOS1-r^{2}SCAN0-2 [248]	R2SCAN0-2	79.37	20.63	50	50	1.333	0.0
SOS1-r^{2}SCAN0-DH [248]	R2SCAN0-I	50	50	87.50	12.50	1.333	0.0
Pr^{2}SCAN50 [248]	PR2SCAN5(	50	50	75	25	1.333	0.0
Pr^{2}SCAN69 [248]	PR2SCAN6!	69.34	30.76	55.56	44.44	1.333	0.0
κPr^{2}SCAN5([248]	KPR2SCAN	50	50	70	30	1.333	0.0
excited state opt.							
SCS/SOS-B2PLYP21 [266]	SCS/ SOS-B2PL:	53	47	73	27	[1]	Page 227, 1
SCS-PBE-QIDH [266]	SCS-PBE-(	69	31	67	33	[2]	Page 227, 2
SOS-PBE-QIDH [266]	SOS-PBE-(	69	31	67	33	[3]	Page 227, 3
SCS-B2GP-PLYP21 [266]	SCS-B2GP-	65	35	64	36	[4]	Page 227, 4

continues on next page

Table 3.8 – continued from previous page

Functional	Keywords	%HFX	%DFX	%GGAC	%MP2C	PS	PT
SOS-B2GP-PLYP21 [266]	SOS-B2GP-	65	35	64	36	[5]	[5]

Range-Separated Double-Hybrid Functionals

Range-separated double-hybrids employ variable amounts of Hartree-Fock exchange much like *range-separated hybrid functionals* and static amounts of *MP2 correlation*. A list of available range-separated double-hybrid functionals is given in Table 3.9.

Hint

For most natively implemented double-hybrids you can simple add the `DLPNO-` prefix to the respective functional simple input keyword, e.g. `DLPNO-WPR2SCAN50`. For all others, the DLPNO treatment can be activated via the `%mp2` block using the `DLPNO true` keyword.

Table 3.9: List of available range-separated double-hybrid functionals. `PS` is the opposite-spin, `PT` the same-spin scaling parameter. μ is the range-separation parameter. Excited-states optimized (ES-Opt.) functionals only employ spin-component-scaling for the CIS(D) doubles correction within the the TD-DFT framework. Same-spin indirect term (CTss), opposite-spin indirect term (CTos), same-spin direct term (CUss), and opposite-spin direct term (CUos) parameters are given in the footnotes.

[1] SCS/SOS-B2PLYP21: CTss: 0.000; CTos: 0.334; CUss: 0.000; CUos: 0.283.
[2] SCS-PBE-QIDH: CTss: 0.070; CTos: 0.515; CUss: 0.096; CUos: 0.524.
[3] SOS-PBE-QIDH: CTss: 0.000; CTos: 0.547; CUss: 0.000; CUos: 0.573.
[4] SCS-B2GP-PLYP21: CTss: 0.018; CTos: 0.475; CUss: 0.000; CUos: 0.468.
[5] SOS-B2GP-PLYP21: CTss: 0.000; CTos: 0.489; CUss: 0.000; CUos: 0.484.

Functional	Keywords	%HFX	μ / bohr^{-1}	%GGAC	%MP2C	PS	PT
ωB97M(2)[6] [256]	WB97M(2)	62.194 – 100	0.30	100	34.096	–	–
ωB97X-2(TQZ) [267]	WB97X-2	63.6158 – 95.1661	0.30	100	100	0.447105	0.529319
ωPr2SCAN50 [248]	WPR2SCAN50	50 – 100	0.214013	65	35	1.333	0.000
RSX-QIDH [268]	RSX-QIDH	69 – 100	0.27	67	33	–	–
RSX-0DH [269]	RSX-0DH	50 – 100	0.33	87.5	12.5	–	–
ωB2PLYP [270]	WB2PLYP	53 – 100	0.30	73	27	–	–
ωB2GP-PLYP [270]	WB2GP-PLYP	65 – 100	0.27	64	36	–	–
ωB88PP86 [270]	WB88PP86	65 – 100	0.20	58	42	–	–
ωPBEPP86 [270]	WPBEPP86	70 – 100	0.18	68	48	–	–
excited state opt.							
SCS/SOS-ωB2PLYP [266]	SCS/SOS-WB2PLYP	53 – 100	0.30	73	27	[7]	[7]
SCS-ωB2GP-PLYP [266]	SCS-WB2GP-PLY	65 – 100	0.27	64	36	[8]	[8]
SOS-ωB2GP-PLYP [266]	SOS-WB2GP-PLY	65 – 100	0.27	64	36	[9]	[9]
SCS-RSX-QIDH [266]	SCS-RSX-QIDH	69 – 100	0.27	67	33	[10]	[10]
SOS-RSX-QIDH [266]	SOS-RSX-QIDH	69 – 100	0.27	67	33	[11]	[11]
SCS-ωB88PP86 [266]	SCS-WB88PP86	65 – 100	0.20	58	42	[12]	[12]
SOS-ωB88PP86 [266]	SOS-WB88PP86	65 – 100	0.20	58	42	[13]	[13]
SCS-ωPBEPP86 [266]	SCS-WPBEPP86	70 – 100	0.18	68	48	[14]	[14]
SOS-ωPBEPP86 [266]	SOS-WPBEPP86	70 – 100	0.18	68	48	[15]	[15]

ωB97M(2)

Some double-hybrid functionals, such as ωB97M(2)[256] must be applied to orbitals converged with a different functional. This can be accomplished with a two-step calculation using `MORead` and `MaxIter=1`. Note that because the orbitals are not obtained self-consistently, only single point energies can be computed in this way, i.e. no density, gradient, or properties! The usage of such functionals may be automatized using the *Compound script language*. An exemplary compound script for a ωB97M(2) single-point calculation is given below:

```
*xyz 0 1
  H 0.0 0.0 0.0
  F 0.0 0.0 0.9
*
# ----------------------------------------------------------------------------
```

(continues on next page)

[6] Must be used with ωB97M-V orbitals (cf. Section 3.3.5).
[7] SCS/SOS-ωB2PLYP: CTss: 0.000; CTos: 0.433; CUss: 0.000; CUos: 0.460.
[8] SCS-ωB2GP-PLYP: CTss: 0.000; CTos: 0.571; CUss: 0.135; CUos: 0.558.
[9] SOS-ωB2GP-PLYP: CTss: 0.000; CTos: 0.570; CUss: 0.000; CUos: 0.610.
[10] SCS-RSX-QIDH: CTss: 0.174; CTos: 0.541; CUss: 0.586; CUos: 0.424.
[11] SOS-RSX-QIDH: CTss: 0.000; CTos: 0.632; CUss: 0.000; CUos: 0.714.
[12] SCS-ωB88PP86: CTss: 0.000; CTos: 0.557; CUss: 0.092; CUos: 0.545.
[13] SOS-ωB88PP86: CTss: 0.000; CTos: 0.559; CUss: 0.000; CUos: 0.599.
[14] SCS-ωPBEPP86: CTss: 0.000; CTos: 0.610; CUss: 0.297; CUos: 0.540.
[15] SOS-ωPBEPP86: CTss: 0.000; CTos: 0.613; CUss: 0.000; CUos: 0.669.

(continued from previous page)

```
↪---
# Energy calculation with the wB97M(2) double-hybrid functional, as defined in
# N. Mardirossian, M. Head-Gordon, J. Chem. Phys. 2018, 148, 241736.
# The final energy is calculated non-variatonally using wB97M-V orbitals.
# Note that FrozenCore is on default - one can turn it off using kw2 =
↪"NoFrozenCore".
# Author: Georgi L. Stoychev
# ------------------------------------------------------------------------------
↪---

# Optional variables:
Variable NLOpt = "SCNL";        # Consistent with Q-Chem. "NL" likely very similar.
↪ Only matters for Step 1.
Variable Basis = "def2-TZVP";   # Should be TZ or higher for reliable results
Variable AuxC  = "def2-TZVP/C"; # RI-MP2 is used by default. Must be compatible␣
↪with the Basis
Variable AuxJ  = "def2/J";      # RIJCOSX is used by default
Variable RIMP2 = True;          # Set to false to turn off RI-MP2
Variable Grid  = "defGrid2";    # Should be OK.
Variable kw1   = "";            # Additional simple input keywords for step 1
Variable kw2   = "";            # Additional simple input keywords for step 2

# Output variables
Variable EwB97MV, EwB97M2;      # Will hold the final energies

# Step 1: wB97M-V calculation to obtain the orbitals
New_Step
  ! wB97M-V &{NLOpt} &{Grid} &{kw1}
  %basis
    Basis "&{Basis}"
    AuxJ  "&{AuxJ}"
  end
Step_End

# Step 2: single iteration with the wB97M(2) functional + MP2 correlation to get␣
↪the final energy
ReadMOs(1);
New_Step
  ! wB97M(2) &{NLOpt} &{Grid} &{kw2} NoIter CalcGuessEnergy
  %basis
    Basis "&{Basis}"
    AuxC  "&{AuxC}"
    AuxJ  "&{AuxJ}"
  end
  %scf
    IgnoreConv 1
  end
  %mp2 RI &{RIMP2} end
Step_End

Read EwB97MV = DFT_Total_En[1];
Read EwB97M2 = MP2_Total_Energy[2];

Print("");
Print(
↪"==============================================================================\
↪n");
Print(" wB97M-V Energy:  %20.10lf\n",EwB97MV);
Print(" wB97M(2) Energy: %20.10lf\n",EwB97M2);

End
```

Customization of Double-Hybrids

As double-hybrid functionals make use of customizable components of *global hybrids* or *range-separated hybrids* they can can be constructed and customized in similar fashion. The MP2 correlation treatment for double-hybrids can be controlled via the `DoMP2 true` and `ScalMP2C` keywords within the `%method` block. Moreover, the MP2 treatment can be further customized via the `%mp2` block. For example, the spin-composite-scaled, range-separated double-hybrid functional ωPr^{2}SCAN50 can be constructed as follows:

```
! RIJCOSX def2-TZVP/C NOTRAH
# wPr2SCAN50
%method
   Method       DFT
   DoMP2        true       # activates MP2 correlation part
   Exchange     X_WR2SCAN
   Correlation  C_R2SCAN
   LDAOpt       C_PWLDA
   ScalHFX      0.50       # fraction of static Hartree-Fock exchange
   ScalDFX      0.00       # fraction of static GGA exchange
   ScalGGAC     0.65       # fraction of GGA correlation
   ScalLDAC     0.65       # fraction of LDA correlation, must be equal to ScalGGAC
   ScalMP2C     0.35       # fraction of MP2 correlation
   RangeSepEXX  true       # activates range-separated exchange
   RangeSepMu   0.214013   # range-separation parameter, should be >0
   RangeSepScal 0.50       # fraction of variable HFX, should sum to 1 with␣
↪ScalHFX and ScalDFX
end
%mp2
   DoSCS        true       # activates spin-component-scaling
   RI           true       # activate RI-MP2
   PS           1.333      # opposite-spin-scaling parameter
   PT           0.000      # same-spin-scaling parameter
end
```

Many other MP2 related options like the DLPNO treatment and ragularization can be found in the *MP2* section of the manual.

Table 3.10: Most important `%method` block input keywords to customize double-hybrid functionals.

Keyword	Options	Description
`DoMP2`	`true`	Activates MP2 correlation part
`ScalMP2C`	`<real>`	Controls the fraction of GGA exchange

Table 3.11: Most important `%mp2` block input keywords to customize double-hybrid functionals. For more details see the MP2 section.

Keyword	Options	Description
`DoSCS`	`true`	Activates spin-component-scaling of MP2 part
`PS`	`<real>`	Controls the opposite-spin-scaling parameter
`PT`	`<real>`	Controls the same-spin-scaling parameter
`RI`	`true`	Activates RI-MP2
`DLPNO`	`true`	Activates DLPNO-MP2

3.3.6 LibXC Functional Library

It is possible to use the functionals provided by LibXC within the ORCA framework[16]. The LibXC version used by ORCA is printed at the beginning of the output. The LibXC reference should be cited when LibXC is used as part of your calculations. For reference, see [271].

The complete list of functionals available via the LibXC interface can always be inspected by typing at the command line

```
orca -libxcfunctionals
```

The list of functionals has the following form:

```
Functionals available via LibXC:
  No.:     ID / Keyword                    - Name
    0:     18 / lda_c_1d_csc               - Casula, Sorella & Senatore
    1:     26 / lda_c_1d_loos              - P-F Loos correlation LDA
    2:     15 / lda_c_2d_amgb              - AMGB (for 2D systems)
    3:     16 / lda_c_2d_prm               - PRM (for 2D systems)
    4:    552 / lda_c_br78                 - Brual & Rothstein 78

   ...
```

The list is grouped by "level" of functional (LDA, GGA, meta-GGA, etc.) and then by part of the energy it models (correlation, exchange, exchange-correlation). Correlation functionals carry a `'_c_'` in their names, exchange functionals an `'_x_'`, and combined exchange-correlation functionals an `'_xc_'`. Additional information for a specific functional can be requested using

```
orca -libxcinfo [functional name]
```

where the functional name is the keyword in the above list.

Specification of LibXC functionals in the ORCA input follows the standard format:

```
%method
  Method     DFT
  Functional hyb_gga_xc_b3lyp
end
```

or in the case of separate exchange and correlation specifications:

```
%method
  Method      DFT
  Exchange    mgga_x_m06_l
  Correlation mgga_c_m06_l
end
```

CAM-type range-separated functionals are supported through the LibXC interface since ORCA 5.0. So are functionals that include non-local (VV10) correlation (see *Non-Local Dispersion Correction (VV10): DFT-NL*). The VV10 part is computed internally by ORCA. Other non-local functionals, such as BEEF-vdW, are not supported. Meta-GGA functionals that depend on the kinetic energy density τ are supported, but not those that depend on the Laplacian of the density $\nabla^2\rho$.

Double-hybrid functionals can be constructed with LibXC components as described in the previous customization sections, but only with separate exchange and correlation components. So `exchange=gga_x_pbe` and `correlation=gga_c_pbe` can be used, but `functional=hyb_gga_xc_pbeh` **cannot** be used in a double-hybrid formulation. Beware that the exchange and correlation contributions calculated by LibXC are simply scaled by `ScalDFX` and `ScalGGAC`, respectively, and no care is taken to separately scale LDA components or alter other internal parameters!

[16] https://libxc.gitlab.io

Screening Thresholds

When the density is smaller than a certain threshold, LibXC skips the evaluation of the functional and instead sets all the output quantities to zero in order to avoid under- and/or overflows. The default thresholds for different functionals are set by the LibXC developers, but may sometimes be too low. We have not performed extensive testing, but allow the user to set the threshold. Similarly, there are thresholds for minimum values of ζ and τ in order to avoid division by zero. The default values are functional-independent and can be changed using the following keywords.

```
%method
  LibXCDensityThreshold 1e-12    # seems to be reasonable
  LibXCZetaThreshold    2e-16    # default value in LibXC
  LibXCTauThreshold     1e-20    # default value in LibXC
end
```

Modifying LibXC Functional Parameters

Starting with ORCA 5.1 it is possible to modify the "external parameters" of a functional that the LibXC interface provides. The available parameters and their default values can be seen in the output or with the `orca -libxcinfo` command above. Please exercise caution when using this interface and if using a published functional reparametrization, make sure you can reproduce results from the original publication. The syntax requires one of the `ExtParamX`, `ExtParamC`, or `ExtParamXC` keywords (depending on which functional is modified), followed by the parameter name in quotation marks, and finally the new value. For example, here is an input for the PWPB95 double-hybrid functional, where the simple input keyword is used to set most parameters (such as the MP2 scaling factors) and only the exchange and correlation parameters of the mPW91 and B95 LibXC functionals, respectively, are modified.

```
! Opt PWPB95 def2-SVP def2-SVP/C

%method
  Exchange    gga_x_mpw91
  Correlation mgga_c_bc95
  ExtParamX   "_bt"     0.00444
  ExtParamX   "_alpha" 19.823391
  ExtParamX   "_expo"   3.7868
  ExtParamC   "_css"    0.03241
  ExtParamC   "_copp"   0.00250
end

*xyz 0 1
  H 0 0 0
  F 0 0 0.9
*
```

Note that the variable definitions in LibXC may be different from the ones used internally in ORCA or in the original publication, due to various constant factors, etc. When in doubt, please consult the LibXC documentation or source code repository — we simply provide access to the interface.

Simple Input of LibXC Functionals

Some LibXC functionals are accessible via the simple input keyword `!LibXC(Keyword)`, e.g. `!LibXC(TPSS)`. The keywords match those of the natively implemented in ORCA. Using this syntax, parameters for the *DFT-D dispersion corrections* are also set automatically (if implemented). Table Table 3.12 lists the available functionals and their LibXC names.

Table 3.12: LibXC functionals available via the simple input `!LibXC(Keyword)`

Keyword	LibXC components	Notes
GGAs		

continues on next page

Table 3.12 – continued from previous page

Keyword	LibXC components	Notes
B97-D	gga_xc_b97_d	Uses D2
B97-D3	gga_xc_b97_d	Uses D3(0)
B97-D4	gga_xc_b97_d	Uses D4
BLYP	gga_x_b88 + gga_c_lyp	
KT2	gga_xc_kt2	
KT3	gga_xc_kt3	
PBE	gga_x_pbe + gga_c_pbe	
REVPBE	gga_x_pbe + gga_c_pbe	κ=1.245
XLYP	gga_xc_xlyp	
meta-GGAs		
B97M-V	mgga_xc_b97m_v	Uses VV10
B97M-D3BJ	mgga_xc_b97m_v	Uses D3(BJ)
B97M-D4	mgga_xc_b97m_v	Uses D4
M06L	mgga_x_m06_l + mgga_c_m06_l	
MN15L	mgga_x_mn15_l + mgga_c_mn15_l	
MTASK-LDA	mgga_x_mtask + lda_c_pw_mod	
MTASK-SCAN	mgga_x_mtask + mgga_c_scan	
REVTM	mgga_x_revtm + mgga_c_revtm	
R2SCAN	mgga_x_r2scan + mgga_c_r2scan	
RSCAN	mgga_x_rscan + mgga_c_rscan	
SCAN	mgga_x_scan + mgga_c_scan	
TASK-CCALDA	mgga_x_task + mgga_c_ccalda	
TASK-CC	mgga_x_task + mgga_c_cc	
TASK-LDA	mgga_x_task + lda_c_pw_mod	
TASK-SCAN	mgga_x_task + mgga_c_scan	
TM	mgga_x_tm + mgga_c_tm	
TPSS	mgga_x_tpss + mgga_c_tpss	
hybrid GGAs		
B1PW	gga_x_b88,gga_c_pw91	Custom mixing
B1PBE	gga_x_b88 + gga_c_pbe	Custom mixing
B3PW91	hyb_gga_xc_b3pw91	Custom mixing
B3LYP	hyb_gga_xc_b3lyp5	
B3LYP/G	hyb_gga_xc_b3lyp	
B3LYPS	hyb_gga_xc_b3lyps	B3LYP*, 15% HFX
B97	hyb_gga_xc_b97	
CAM-B3LYP	hyb_gga_xc_cam_b3lyp	Custom mixing
CASE21	hyb_gga_xc_case21	
G1LYP	gga_x_g96 + gga_c_lyp	Custom mixing
HSE06	hyb_gga_xc_hse06	
HSE12	hyb_gga_xc_hse12	
HSE12S	hyb_gga_xc_hse12s	
HSESOL	hyb_gga_xc_hse_sol	
KMLYP	hyb_gga_xc_kmlyp	
LB07	hyb_gga_xc_lb07	
LC-BLYP	hyb_gga_xc_lc_blyp	Custom mixing
MPW1LYP	hyb_gga_xc_mpw1lyp	Custom mixing
MPW1PW	hyb_gga_xc_mpw1pw	Custom mixing
PBE0	hyb_gga_xc_pbeh	
PBE38	hyb_gga_xc_pbe38	
PBE50	hyb_gga_xc_pbe50	
PW91_0	gga_x_pw91 + gga_c_pw91	Custom mixing
PW1PW	gga_x_pw91 + gga_c_pw91	Custom mixing
REVPBE0	gga_x_pbe,gga_c_pbe	κ=1.245
REVPBE38	gga_x_pbe,gga_c_pbe	37.5% HFX, κ=1.245
WHPBE0	hyb_gga_xc_whpbe0	

continues on next page

Table 3.12 – continued from previous page

Keyword	LibXC components	Notes
WB97	hyb_gga_xc_wb97	
WB97X	hyb_gga_xc_wb97x	
WB97X-D3	hyb_gga_xc_wb97x_d3	Uses D3(0)
WB97X-V	hyb_gga_xc_wb97x_v	Uses VV10
WB97X-D3BJ	hyb_gga_xc_wb97x_v	Uses D3(BJ)
WB97X-D4	hyb_gga_xc_wb97x_v	Uses D4
WB97X-D4REV	hyb_gga_xc_wb97x_v	Uses D4 with revised parameters
hybrid meta-GGAs		
GAS22	hyb_mgga_xc_gas22	
M05	hyb_mgga_x_m05 + mgga_c_m05	
M052X	hyb_mgga_x_m05_2x + mgga_c_m05_2x	
M06	hyb_mgga_x_m06 + mgga_c_m06	
M062X	hyb_mgga_x_m06_2x + mgga_c_m06_2x	
M06SX	hyb_mgga_x_m06_sx + mgga_c_m06_sx	
MN15	hyb_mgga_x_mn15 + mgga_c_mn15	
LC-TMLYP	hyb_mgga_xc_lc_tmlyp	
PW6B95	hyb_mgga_xc_pw6b95	
R2SCANH	hyb_mgga_xc_r2scanh	
R2SCAN0	hyb_mgga_xc_r2scan0	
R2SCAN38	hyb_mgga_xc_r2scanh	37.5% HFX
R2SCAN50	hyb_mgga_xc_r2scan50	
SCAN0	hyb_mgga_x_scan0 + mgga_c_scan	
TPSS0	hyb_mgga_xc_tpss0	
TPSSH	hyb_mgga_xc_tpssh	
WB97M-V	hyb_mgga_xc_wb97m_v	Uses VV10
WB97M-D3BJ	hyb_mgga_xc_wb97m_v	Uses D3(BJ)
WB97M-D4	hyb_mgga_xc_wb97m_v	Uses D4
WB97M-D4REV	hyb_mgga_xc_wb97m_v	Uses D4 with revised parameters
double-hybrids		
B2PLYP	gga_x_b88 + gga_c_lyp	Custom mixing
B2GP-PLYP	gga_x_b88 + gga_c_lyp	Custom mixing
B2T-PLYP	gga_x_b88 + gga_c_lyp	Custom mixing
B2K-PLYP	gga_x_b88 + gga_c_lyp	Custom mixing
B2NC-PLYP	gga_x_b88 + gga_c_lyp	Custom mixing
DSD-PBEB95-D3	gga_x_pbe + mgga_c_bc95	Custom mixing, uses D3(BJ)
DSD-PBEB95-D4	gga_x_pbe + mgga_c_bc95	Custom mixing, uses D4
MPW2-PLYP	gga_x_mpw91 + gga_c_lyp	Custom mixing
PWPB95	gga_x_mpw91 + mgga_c_bc95	Custom mixing and ExtParam's
PBE0-DH	gga_x_pbe + gga_c_pbe	Custom mixing
PBE0-2	gga_x_pbe + gga_c_pbe	Custom mixing
PBE-QIDH	gga_x_pbe + gga_c_pbe	Custom mixing
PR2SCAN50	mgga_x_r2scan + mgga_c_r2scan	Custom mixing
PR2SCAN69	mgga_x_r2scan + mgga_c_r2scan	Custom mixing
SOS1-R2SCAN0-DH	mgga_x_r2scan + mgga_c_r2scan	Custom mixing
SOS1-R2SCAN-CIDH	mgga_x_r2scan + mgga_c_r2scan	Custom mixing
SOS1-R2SCAN-QIDH	mgga_x_r2scan + mgga_c_r2scan	Custom mixing
SOS1-R2SCAN0-2	mgga_x_r2scan + mgga_c_r2scan	Custom mixing

3.4 Dispersion Corrections

A major weakness of conventional density functional theory (and Hartee-Fock) is the lacking description of long-range correlation effects including London dispersion. London dispersion represents the attractive component of the van-der-Waals interaction and even though it is a rather weak interaction, its correct description can be crucial for many cases. This is specifically the case for non-covalent interactions and extended molecules. However, there are several well-established dispersion-corrections available to largely correct for this shortcoming. Within ORCA various options are available that include Grimme's D3 [272, 273] and D4 [159, 274] corrections as well as the non-local variant of VV10 (NL, [275, 276]) are available.

Some notes on dispersion corrections

- Dispersion corrections do not only affect non-covalent complexes, but also conformational energies and structures which are heavily influenced by intramolecular dispersion. Therefore, including the dispersion correction is almost always recommended or even required (except for a handful of cases where it cannot, should not or need not be used, see below).
- DFT calculations with small basis sets (such as double-ζ basis sets) often yield attractive potential energy surfaces even without the dispersion correction. However, this is due to basis set superposition error (BSSE), and the interaction energy brought about by the BSSE frequently does not match the true interaction energy due to dispersion (because they have completely different origins). Therefore, although a DFT double zeta calculation without the dispersion correction may appear to give qualitatively correct results, or occasionally even better results than a double zeta calculation with dispersion corrections (because in the latter case one typically overestimates the total attraction), it is still highly recommended to "get the right answer for the right reason" by reducing the BSSE and turning on the dispersion correction. The BSSE can be corrected by a variety of means, for example (1) by using a larger basis set; (2) by using the counterpoise correction (*Counterpoise Corrections*); or (3) by using the geometrical counterpoise correction (section *Geometrical Counterpoise Correction (gCP)*). Of these, (3) is available at almost no cost (including analytic gradient contributions), and is especially suitable for geometry optimization of large molecules. Otherwise (1) (or its combination with (2)) may be more appropriate due to its higher accuracy.
- Functionals that contain VV10-type non-local dispersion (in general, these are the functionals whose names end with "-V") do not need (and cannot be used together with) dispersion corrections. The same holds for post-HF and multireference methods, like MP2, CCSD(T), CASSCF and NEVPT2. However, one can add a dispersion correction on top of HF.
- Certain functionals, especially the Minnesota family of functionals (e.g. M06-2X), describe medium-range dispersion but miss long-range dispersion. They give reasonable dispersion energies for small to medium systems but may slightly underestimate the dispersion energies for large systems. As the uncorrected functional already accounts for the bulk of the dispersion in this case, the dispersion correction is much less important than e.g. the case of B3LYP, and should in general be considered as beneficial but not mandatory.

3.4.1 Grimme's DFT-D3 and DFT-D4

DFT-D3 and DFT-D4 are atom-pairwise (atom-triplewise) dispersion corrections which can be added to the KS-DFT energies and gradient [272]. In ORCA, D3 is available for all elements up to Pu (Z = 94) and D4 up to Lr (Z = 103).

$$E_{\text{DFT-D}} = E_{\text{KS-DFT}} + E_{\text{disp}} \tag{3.9}$$

E_{disp} is then the sum of the two- and three-body contributions to the dispersion energy, $E_{\text{disp}} = E^{(2)} + E^{(3)}$. Most important is the two-body term, which is given at long range by:

$$E_{\text{disp}} = -\sum_{A<B}\sum_{n=6,8} s_n \frac{C_n^{AB}}{r_{AB}^n} \tag{3.10}$$

where C_n^{AB} denotes the averaged (isotropic) n^{th}-order dispersion coefficient for atom pair *AB* and r_{AB} is their internuclear distance. s_n is a functional-dependent scaling factor (see below). In the general case, an adequate damping

function must be employed. The more recent D4 model basically uses a similar theory but further introduces an atomic partial charge dependence of the atomic reference polarizabilities.

Damping Functions

In order to avoid near-singularities for small r_{AB}, the dispersion contribution needs to be damped at short distances. One possible way is to use rational damping as proposed by Becke and Johnson [277, 278, 279]:

$$E^{(2)} = -\sum_{A<B}\sum_{n=6,8} s_n \frac{C_n^{AB}}{r_{AB}^n + f(R_0^{AB})^n} \tag{3.11}$$

with [279]

$$R_0^{AB} = \sqrt{\frac{C_8^{AB}}{C_6^{AB}}} \tag{3.12}$$

and

$$f(R_0^{AB}) = a_1 R_0^{AB} + a_2. \tag{3.13}$$

Damping the dispersion contribution to zero for short ranges (as in Ref. [272]) is also possible:

$$E^{(2)} = -\sum_{A<B}\sum_{n=6,8} s_n \frac{C_n^{AB}}{r_{AB}^n} f_{d,n}(r_{AB}) \tag{3.14}$$

with

$$f_{d,n} = \frac{1}{1 + 6(\frac{r_{AB}}{s_{r,n} R_0^{AB}})^{-\alpha_n}} \tag{3.15}$$

Note that the R_0^{AB} used with this damping are from Ref. [272]. For more information on the supported damping functions, see Ref [273].

Three-body term

It is possible to calculate the three-body dispersion contributions according to

$$E^{(3)} = -\sum_{A<B<C} \frac{C_9^{ABC}(3\cos\theta_a\cos\theta_b\cos\theta_c + 1)}{(r_{AB} r_{BC} r_{CA})^3} f_{d,(3)}(\overline{r}_{ABC}) \tag{3.16}$$

where θ_a, θ_b and θ_c are the internal angles of the triangle formed by r_{AB}, r_{BC} and r_{CA}. The C_9 coefficient is approximated by

$$C_9^{ABC} \approx -\sqrt{C_6^{AB} C_6^{AC} C_6^{BC}} \tag{3.17}$$

The three-body contribution has a small effect on medium-sized molecules and is damped according to equation (3.16). The damping function $f_{d,(3)}(\overline{r}_{ABC})$ is similar to the one shown in equation (3.15) with $\overline{r}_{ABC}$ being the geometric mean of r_{AB}, r_{BC} and r_{CA}. It can be used with both variants of the $E^{(2)}$ term, although the three-body term itself will always be calculated using the zero damping scheme. Adding the three-body correction has proven to give quite accurate results for the thermochemistry of supramolecular systems[280]. Note that the three-body term is included in D4 by default.

Basic Usage

The D3 correction can be invoked by the `!D3` keyword that will automatically make use of the default Becke-Johnson damping and is thus equivalent to `!D3BJ`. The zero damping scheme can be used with `!D3ZERO`. Note that the Becke-Johnson scheme is generally recommended.

```
!PBE D3
```

The three-body term can be invoked via the keyword `!ABC`.

```
!PBE D3 ABC
```

Note

Note that not all methods are parameterized for all damping options. E.g. Hartree-Fock (HF) is only parametrized with BJ-damping, while M06 is only parameterized with zero damping.

The more recent and generally recommended D4 correction can be used with the `!D4` keyword. Note that D4 uses Becke-Johnson damping and the three-body term by default.

```
!PBE D4
```

The D4 correction can also be defined in the `%method` block of the input:

```
! PBE def2-TZVP

%method
  DFTDOpt      5      # 5 = D4; 4 = D3(BJ)
end
```

Custom parameters

If there is a functional for which parameters are known but not yet implemented into ORCA, they can be sepcified manually via the `%method` block.

Warning

Usage of custom parameters is at own risk and is generally not recommended. A list of some available parameters for D3 can be found at the website of the Grimme group.

Important

GGA and hybrid functionals should only be used with $s_6 = 1.0$ to ensure asymptotically correct behavior. Double-hybrid functionals already account for parts of the dispersion interaction and should therefore be used with $s_6 < 1.0$.

For D3 with Becke-Johnson damping, the s_6, a_1, s_8, and a_2 parameters can be adjusted:

```
!B2PLYP D3

%method
  D3S6 0.6400
  D3S8 0.9147
  D3A1 0.3065
```

(continues on next page)

(continued from previous page)

```
  D3A2 5.0570
end
```

For D4, the s_9 parameter of the three-body term can be adjusted as well.

```
!B2PLYP D4

%method
  D4S6 0.6400
  D4S8 1.1689
  D4A1 0.4415
  D4A2 4.7311
  D4S9 1.0000
end
```

For the D3 variant with zero damping the parameters s_6, rs_6, s_8, and α_6 can be adjusted.

```
!BLYP D3ZERO

%method
  D3S6     1.000
  D3RS6    1.094
  D3S8     1.682
  D3alpha6 14
end
```

For some functionals, there are also special parameters, which were optimized for use with the def2-TZVPP triple-zeta basis set. This option is only available with zero damping and can be invoked by the keywords `!D3ZERO D3TZ`. Preliminary results in the SI of Ref. [272] indicate that results are only slightly worse than with quadruple-zeta basis sets using the default parameters. This option should be carefully tested for future use in large computations.

Example

In this first example, the DFT-D4 dispersion correction is used.

```
!PBE def2-TZVP D4

* xyz 0 1
  C     0.000000     0.000000     0.000000
  O     0.000000     0.000000     1.400000
  O     0.000000     0.000000    -1.400000
*
```

The output for the D4 dispersion correction in the ORCA output will look like this:

```
-------------------------------------------------------------------------------
                          DFT DISPERSION CORRECTION

                                DFTD4 V3.4.0
-------------------------------------------------------------------------------
The PBE functional is recognized
Active option DFTDOPT                   ...         5

-------------------------   ----------------
Dispersion correction           -0.000936618
-------------------------   ----------------


-------------------------   --------------------
```

(continues on next page)

(continued from previous page)

```
FINAL SINGLE POINT ENERGY      -188.360452990031
-------------------------   --------------------
```

The total dispersion energy E_{disp} is given as the `Dispersion correction` and is automatically included in the final single point energy. As discussed above, the individual parameters may be adjusted manually.

```
!PBE def2-TZVP D4

%method
  D4S6 1.0
  D4A1 0.4289
  D4S8 0.7875
  D4A2 4.4407
  D4S9 0.9000
end

* xyz 0 1
  C     0.000000     0.000000     0.000000
  O     0.000000     0.000000     1.400000
  O     0.000000     0.000000    -1.400000
*
```

This results in the same output as above, but with additional messages that user inputs were found for the parameters:

```
A user input s6-coefficient scaling factor has been recognized
A user input a1-coefficient scaling factor has been recognized
A user input s8-coefficient scaling factor has been recognized
A user input a2-coefficient scaling factor has been recognized
A user input s9-coefficient scaling factor has been recognized
```

Keywords

Table 3.13: Simple input keywords for the DFT-D corrections.

Keyword	Description
`D2`	Activates D2 correction
`D3`	Activates D3 correction with BJ damping (D3(BJ))
`D3BJ`	Activates D3 correction with BJ damping (D3(BJ))
`D3ZERO`	Activates D3 correction with zero damping (D3(0))
`D30`	Activates D3 correction with zero damping (D3(0)), equivalent to `D3ZERO`
`D3TZ`	Activates special triple-ζ D3 parameterization (only with `D3ZERO`)
`D4`	Activates D4 correction (BJ and ATM included)
`ABC`	Activates three-body term
`ATM`	Activates three-body term, equivalent to `ABC`
`NOVDW`	Deactivates DFT-D dispersion corrections

Table 3.14: `%method` block input keywords for the DFT-D corrections.

Keyword	Options	Description
DFTDOPT	1	Activate the original D correction[281] (deprecated)
	2	Activate the D2 correction (deprecated)
	3	Activate the D3ZERO correction
	4	Activate the D3BJ correction
	5	Activate the D4 correction
D3S6	<real>	Controls s_6 parameter for D3
D3S8	<real>	Controls s_8 parameter for D3
D3A1	<real>	Controls a_1 parameter for D3
D3A2	<real>	Controls a_2 parameter for D3
D3RS6	<real>	Controls rs_6 parameter for zero damping
D3alpha6	<real>	Controls α_6 parameter for zero damping
D4S6	<real>	Controls s_6 parameter for D4
D4S8	<real>	Controls s_8 parameter for D4
D4A1	<real>	Controls a_1 parameter for D4
D4A2	<real>	Controls a_2 parameter for D4
D4S9	<real>	Controls s_9 parameter of ABC term for D4
DFTDScaleC6	<real>	Defines the C6 scaling coefficient

Available Parameters

Table 3.15: DFT-D4 (!D4) parameters used for various functionals.

Functional	s_6	s_8	a_1	a_2	s_9
HF	1.0000	1.61679827	0.44959224	3.35743605	1.0000
BLYP	1.0000	2.34076671	0.44488865	4.09330090	1.0000
BPBE	1.0000	3.64405246	0.52905620	4.11311891	1.0000
BP86	1.0000	3.35497927	0.43645861	4.92406854	1.0000
BPW91	1.0000	3.24571506	0.50050454	4.12346483	1.0000
GLYP	1.0000	4.23798924	0.38426465	4.38412863	1.0000
LB94	1.0000	2.59538499	0.42088944	3.28193223	1.0000
mPWLYP	1.0000	1.25842942	0.25773894	5.02319542	1.0000
mPWPW	1.0000	1.82596836	0.34526745	4.84620734	1.0000
OLYP	1.0000	2.74836820	0.60184498	2.53292167	1.0000
OPBE	1.0000	3.06917417	0.68267534	2.22849018	1.0000
PBE	1.0000	0.95948085	0.38574991	4.80688534	1.0000
RPBE	1.0000	1.31183787	0.46169493	3.15711757	1.0000
REVPBE	1.0000	1.74676530	0.53634900	3.07261485	1.0000
PW86PBE	1.0000	1.21362856	0.40510366	4.66737724	1.0000
RPW86PBE	1.0000	1.12624034	0.38151218	4.75480472	1.0000
PW91	1.0000	0.77283111	0.39581542	4.93405761	1.0000
PW91P86	1.0000	0.32801227	0.35874687	6.05861168	1.0000
XLYP	1.0000	1.62972054	0.11268673	5.40786417	1.0000
B97BECKE	1.0000	0.87854260	0.29319126	4.51647719	1.0000
TPSS	1.0000	1.76596355	0.42822303	4.54257102	1.0000
revTPSS	1.0000	1.53089454	0.44880597	4.64042317	1.0000
SCAN	1.0000	1.46126056	0.62930855	6.31284039	1.0000
B1LYP	1.0000	1.98553711	0.39309040	4.55465145	1.0000
B3LYP	1.0000	2.02929367	0.40868035	4.53807137	1.0000
BHANDHLYP	1.0000	1.65281646	0.27263660	5.48634586	1.0000

continues on next page

Table 3.15 – continued from previous page

Functional	s_6	s_8	a_1	a_2	s_9
B1P	1.0000	3.36115015	0.48665293	5.05219572	1.0000
B3P86	1.0000	3.08822155	0.47324238	4.98682134	1.0000
B1PW91	1.0000	3.02227550	0.47396846	4.49845309	1.0000
B3PW91	1.0000	2.88364295	0.46990860	4.51641422	1.0000
O3LYP	1.0000	1.75762508	0.10348980	6.16233282	1.0000
REVPBE0	1.0000	1.57185414	0.38705966	4.11028876	1.0000
REVPBE38	1.0000	1.66597472	0.39476833	4.39026628	1.0000
PBE0	1.0000	1.20065498	0.40085597	5.02928789	1.0000
PWP1	1.0000	0.60492565	0.46855837	5.76921413	1.0000
PW1PW	1.0000	0.96850170	0.42427511	5.02060636	1.0000
mPW1PW	1.0000	1.80841716	0.42961819	4.68892341	1.0000
mPW1LYP	1.0000	1.15591153	0.25603493	5.32083895	1.0000
PW6B95	1.0000	-0.31926054	0.04142919	5.84655608	1.0000
TPSSh	1.0000	1.85897750	0.44286966	4.60230534	1.0000
TPSS0	1.0000	1.62438102	0.40329022	4.80537871	1.0000
X3LYP	1.0000	1.54701429	0.20318443	5.61852648	1.0000
M06L	1.0000	0.59493760	0.71422359	6.35314182	1.0000
M06	1.0000	0.16366729	0.53456413	6.06192174	1.0000
wB97	1.0000	6.55792598	0.76666802	8.36027334	1.0000
wB97X	1.0000	-0.07519516	0.45094893	6.78425255	1.0000
B97M-D4	1.0000	0.66330000	0.42880000	3.99350000	1.0000
CAM-B3LYP	1.0000	1.66041301	0.40267156	5.17432195	1.0000
LC-BLYP	1.0000	1.60344180	0.45769839	7.86924893	1.0000
B2PLYP	0.6400	1.16888646	0.44154604	4.73114642	1.0000
B2GP-PLYP	0.5600	0.94633372	0.42907301	5.18802602	1.0000
mPW2PLYP	0.7500	0.45788846	0.42997704	5.07650682	1.0000
PWPB95	0.8200	-0.34639127	0.41080636	3.83878274	1.0000
B97-D	1.0000	1.69460052	0.28904684	4.13407323	1.0000
RSCAN	1.0000	0.87728975	0.49116966	5.75859346	1.0000
r2SCAN	1.0000	0.60187490	0.51559235	5.77342911	1.0000
r2SCANh	1.0000	0.83240000	0.49440000	5.90190000	1.0000
r2SCAN0	1.0000	0.89920000	0.47780000	5.87790000	1.0000
r2SCAN50	1.0000	1.04710000	0.45740000	5.89690000	1.0000
wB97X-D4rev	1.0000	0.44850000	0.33060000	4.27900000	1.0000
wB97M-D4rev	1.0000	0.84200000	0.35900000	4.66800000	1.0000
wR2SCAN	1.0000	1.00000000	0.38340000	5.78890000	1.0000
r2SCAN0-DH	0.9424	0.38560000	0.42710000	5.85650000	1.0000
r2SCAN-CIDH	0.8666	0.53360000	0.41710000	5.91250000	1.0000
r2SCAN-QIDH	0.7867	0.29550000	0.40010000	5.83000000	1.0000
r2SCAN0-2	0.7386	0.00000000	0.40300000	5.51420000	1.0000
Pr2SCAN50	0.7964	0.34210000	0.46630000	5.79160000	1.0000
kPr2SCAN50	0.8402	0.12120000	0.43820000	5.82320000	1.0000
wPr2SCAN50	0.8143	0.38420000	0.41350000	5.87730000	1.0000
Pr2SCAN69	0.7167	0.00000000	0.46440000	5.25630000	1.0000
revDSD-PBEP86/2021	0.5917	0.00000000	0.37100000	4.20140000	1.0000
revDOD-PBEP86/2021	0.6158	0.00000000	0.34400000	4.24270000	1.0000
LRC-PBE	1.0000	1.17000000	0.37800000	4.81600000	1.0000

Table 3.16: DFT-D3(BJ) (`!D3BJ`) parameters used for various functionals.

Functional	s_6	s_8	a_1	a_2	s_9
HF	1.0000	0.91710000	0.33850000	2.88300000	1.0000
BP86	1.0000	3.28220000	0.39460000	4.85160000	1.0000

continues on next page

Table 3.16 – continued from previous page

Functional	s_6	s_8	a_1	a_2	s_9
BLYP	1.0000	2.69960000	0.42980000	4.23590000	1.0000
REVPBE	1.0000	2.35500000	0.52380000	3.50160000	1.0000
B97-D	1.0000	2.26090000	0.55450000	3.22970000	1.0000
PBE	1.0000	0.78750000	0.42890000	4.44070000	1.0000
RPBE	1.0000	0.83180000	0.18200000	4.00940000	1.0000
RPW86PBE	1.0000	1.38450000	0.46130000	4.50620000	1.0000
B3LYP	1.0000	1.98890000	0.39810000	4.42110000	1.0000
BHANDHLYP	1.0000	1.03540000	0.27930000	4.96150000	1.0000
TPSS	1.0000	1.94350000	0.45350000	4.47520000	1.0000
TPSS0	1.0000	1.25760000	0.37680000	4.58650000	1.0000
PBE0	1.0000	1.21770000	0.41450000	4.85930000	1.0000
REVPBE38	1.0000	1.47600000	0.43090000	3.94460000	1.0000
PW6B95	1.0000	0.72570000	0.20760000	6.37500000	1.0000
B2PLYP	0.6400	0.91470000	0.30650000	5.05700000	1.0000
mPWLYP	1.0000	2.00770000	0.48310000	4.53230000	1.0000
OLYP	1.0000	2.62050000	0.52990000	2.80650000	1.0000
BPBE	1.0000	4.07280000	0.45670000	4.39080000	1.0000
OPBE	1.0000	3.38160000	0.55120000	2.94440000	1.0000
B3PW91	1.0000	2.85240000	0.43120000	4.46930000	1.0000
REVPBE0	1.0000	1.75880000	0.46790000	3.76190000	1.0000
TPSSh	1.0000	2.23820000	0.45290000	4.65500000	1.0000
CAM-B3LYP	1.0000	2.06740000	0.37080000	5.47430000	1.0000
B2GP-PLYP	0.5600	0.25970000	0.00000000	6.33320000	1.0000
PWPB95	0.8200	0.29040000	0.00000000	7.31410000	1.0000
SCAN	1.0000	0.00000000	0.53800000	5.42000000	1.0000
RSCAN	1.0000	1.08860000	0.47020000	5.73410000	1.0000
R2SCAN	1.0000	0.78980000	0.49480000	5.73080000	1.0000
R2SCANh	1.0000	1.12360000	0.47090000	5.91570000	1.0000
R2SCAN0	1.0000	1.18460000	0.45340000	5.89720000	1.0000
R2SCAN50	1.0000	1.32940000	0.43110000	5.92400000	1.0000
WR2SCAN	1.0000	1.00000000	0.38340000	5.78890000	1.0000
r2SCAN0-DH	0.9424	0.38560000	0.42710000	5.85650000	1.0000
r2SCAN-CIDH	0.8666	0.53360000	0.41710000	5.91250000	1.0000
r2SCAN-QIDH	0.7867	0.29550000	0.40010000	5.83000000	1.0000
r2SCAN0-2	0.7386	0.00000000	0.40300000	5.51420000	1.0000
Pr2SCAN50	0.7964	0.34210000	0.46630000	5.79160000	1.0000
kPr2SCAN50	0.8402	0.12120000	0.43820000	5.82320000	1.0000
wPr2SCAN50	0.8143	0.38420000	0.41350000	5.87730000	1.0000
Pr2SCAN69	0.7167	0.00000000	0.46440000	5.25630000	1.0000
revDSD-PBEP86/2021	0.5917	0.00000000	0.37100000	4.20140000	1.0000
revDOD-PBEP86/2021	0.6158	0.00000000	0.34400000	4.24270000	1.0000
DSD-BLYP	0.5000	0.21300000	0.00000000	6.05200000	1.0000
DSD-BLYP/2013	0.5700	0.00000000	0.00000000	5.40000000	1.0000
DSD-PBEB95	0.6100	0.00000000	0.00000000	6.20000000	1.0000
DSD-PBEP86	0.4180	0.00000000	0.00000000	5.65000000	1.0000
DSD-PBEP86/2013	0.4800	0.00000000	0.00000000	5.60000000	1.0000
B97M-D3BJ	1.0000	0.13840000	-0.07800000	5.59460000	1.0000
wB97X-D3BJ	1.0000	0.26410000	0.00000000	5.49590000	1.0000
wB97M-D3BJ	1.0000	0.39080000	0.56600000	3.12800000	1.0000
wB97X-2	0.5470	0.00000000	3.52000000	7.79500000	1.0000
PBE0DH	0.8800	1.08900000	0.00000000	6.38500000	1.0000
PBE02	0.5400	0.51500000	0.00000000	8.34500000	1.0000
PBE-QIDH	0.6100	0.56600000	0.11400000	7.53800000	1.0000

Table 3.17: DFT-D3(0) (`!D3ZERO`) parameters used for various functionals.

Functional	s_6	rs_6	s_8	α_6
HF	1.0000	1.1580	1.7460	14
BLYP	1.0000	1.0940	1.6820	14
BP86	1.0000	1.1390	1.6830	14
B97-D	1.0000	0.8920	0.9090	14
REVPBE	1.0000	0.9230	1.0100	14
PBE	1.0000	1.2170	0.7220	14
RPBE	1.0000	0.8720	0.5140	14
TPSS	1.0000	1.1660	1.1050	14
B3LYP	1.0000	1.2610	1.7030	14
PBE0	1.0000	1.2870	0.9280	14
PW6B95	1.0000	1.5230	0.8620	14
TPSS0	1.0000	1.2520	1.2420	14
B2PLYP	0.6400	1.4270	1.0220	14
B2GP-PLYP	0.5600	1.5860	0.7600	14
PWPB95	0.8200	1.5570	0.7050	14
mPWLYP	1.0000	1.2390	1.0980	14
BPBE	1.0000	1.0870	2.0330	14
BHANDHLYP	1.0000	1.3700	1.4420	14
TPSSh	1.0000	1.2230	1.2190	14
REVPBE0	1.0000	0.9490	0.7920	14
REVPBE38	1.0000	1.0210	0.8620	14
RPW86PBE	1.0000	1.2240	0.9010	14
B3PW91	1.0000	1.1760	1.7750	14
M06L	1.0000	1.5810	0.0000	14
M06	1.0000	1.3250	0.0000	14
M062X	1.0000	1.6190	0.0000	14
wB97X-D3	1.0000	1.2810	1.0000	14
CAM-B3LYP	1.0000	1.3780	1.2170	14
SCAN	1.0000	1.3240	0.0000	14
wB97X-2	0.5470	4.7330	-0.1090	14
PBE0DH	0.8800	1.1280	0.0000	14
PBE02	0.5400	1.2430	-0.2280	14
PBE-QIDH	0.4000	1.1140	0.0540	14

3.4.2 Non-Local Dispersion Correction (VV10): DFT-NL

A different route to correct for London dispersion is followed by the van der Waals Density Functional (vdW-DF) as pioneered by Langreth and Lundquist [282]. These methods use only the electron density to include such dispersion/correlation effects and can be used in a self-consistent way also correction the molecular orbitals and not only energies. The vdW correlation functional VV10 of Vydrov and Van Voorhis [275] currently seems to be the most promising candidate for a general and accurate electronic structure method.

We use the term DFT-NL for any density functional in combination with the non-local part of the VV10 functional with an optimized parameter b, which will be defined below. The performance of these methods has been evaluated in Ref. [276] using the GMTKN30 [250, 283, 284] database and the S66 set [285]. The performance of weak hydrogen bonds were evaluated in Ref. [286].

DFT-NL and DFT-D3/D4 perform very similarly, but NL is to be preferred for metallic systems or when the basic electronic structure changes significantly (e.g. oxidations or ionizations). As a recent example, Janes and Iron showed that for functionals such as wB97X-V, including VV10 correlation results in very high quality reaction barriers when metals are involved [287].

The total exchange-correlation (XC) energy of VV10-type functionals is defined in eq. (3.18). It is composed of standard exchange (X) and correlation (C) parts and the non-local (NL) term, which covers (mainly) long-range

dispersive energy:

$$E_{\text{XC}}^{\text{DFT-NL}} = E_{\text{X}}^{\text{(hybrid)GGA}} + E_{\text{C}}^{\text{GGA}} + E_{\text{C-NL}}^{\text{VV10}} \tag{3.18}$$

The NL term is given by the following double integral:

$$E_{\text{C-NL}}^{\text{VV10}} = \int dr\rho(r)\left[\beta + \frac{1}{2}\int dr'\rho(r')\varphi(r,r')\right] \tag{3.19}$$

where ρ is the total electron density, and the definitions of the kernel $\varphi(r,r')$ and β are as follows (in a.u.):

$$\varphi\left(r,r'\right) = -\frac{3}{2gg'\left(g+g'\right)} \tag{3.20}$$

$$\beta = \frac{1}{32}\left[\frac{3}{b^2}\right]^{3/4} \tag{3.21}$$

with

$$\begin{aligned} g\left(r\right) &= \omega_0\left(r\right)R^2 + \kappa(r) \\ R &= |r-r'| \\ \omega_0\left(r\right) &= \sqrt{C\left|\frac{\nabla\rho(r)}{\rho(r)}\right|^4 + \frac{4\pi}{3}\rho(r)} \\ \kappa\left(r\right) &= b\frac{3\pi}{2}\left[\frac{\rho(r)}{9\pi}\right]^{1/6} \end{aligned} \tag{3.22}$$

In the original definition, the short-range attenuation parameter b appearing in κ and β was fitted to the S22 set [288] of non-covalent interactions ($b = 5.9$ for the rPW86PBE GGA). The other parameter $C = 0.0093$, appearing in ω_0, determines the long-range behavior, and was set to its original value. Other DFT-NL functionals are constructed analogously. For a detailed discussion of the derivation of the formulas and their physical meaning and basis, see the references given above.

The defined energy of the non-local DFT-NL exchange-correlation functional can be computed in two ways: as a post-SCF addition based on a converged SCF density or in a self-consistent treatment. In our implementation of the non-self-consistent DFT-NL functional, a self-consistent DFT computation is performed as the first step. In the second step, the optimized electron density from the DFT computation is taken as input for the energy calculation of the non-local part. In the self-consistent treatment, the orbitals and density are optimized in the presence of the full DFT + VV10 exchange-correlation potential. According to many test calculations, an self-consistent treatment is rarely necessary for normal energy evaluations. The computation of the double integral given in eq. (3.19) requires using an integration grid, just like for normal exchange-correlation functionals. This grid is built similarly to the regular grids available in the ORCA, see Sec. *Numerical Integration*.

Basic usage

The non-self-consistent variant of NL can be invoked by the `!NL` keyword.

```
!B3LYP NL
```

The self-consistent variant by `!SCNL`.

```
!B3LYP SCNL
```

In the current version of ORCA, NL parameters are implemented for various functionals and HF:

Table 3.18: Methods with available NL parameters. If no values are given, the default parameters are used.

Method	b	C	$NLScal$
Hartree-Fock			
HF	3.9		
(meta-)GGA			
BLYP	4.0		
BP86	4.4		
PBE	6.4		
REVPBE	3.7		
RPBE	4.0		
RPW86PBE	5.9		
SCANfunc	6.4		
RSCAN	10.8		
R2SCAN	12.3		
TPSS	5.0		
B97M-V	6.0	0.01	
Hybrid			
B3LYP	4.8		
B3LYP/G	4.8		
B3P86	5.3		
B3PW91	4.5		
mPW1PW	5.3		
PBE0	6.9		
PW1PW	7.7		
PW6B95	9.0		
REVPBE0	4.3		
REVPBE38	4.7		
TPSSh	5.2		
TPSS0	5.5		
R2SCANh (see [238])	11.9		
R2SCAN0 (see [238])	11.4		
R2SCAN50 (see [238])	10.8		
Range-separated hybrid			
LRC-PBE	6.4		
WR2SCAN (see [248])	9.2612		
WB97X-V	6.0	0.01	
WB97M-V	6.0	0.01	
Double-hybrid			
B2PLYP (see [289])	7.8		
DSD-BLYP (see [290])	12.0		
DSD-PBEP86 (see [290])	14.2		
DSD-BLYP/2013	12.0		
DSD-PBEP86/2013	14.2		
PWPB95 (see [290])	11.1		
PR2SCAN50 (see [248])	10.9207		0.75
KPR2SCAN50 (see [248])	10.6723		0.70
PR2SCAN69 (see [248])	9.0691		0.5556
Range-separated double-hybrid			
WPR2SCAN50 (see [248])	9.4149		0.65
WB97M(2)	10.0	0.01	0.65904

Important

Head-Gordon's ωB97X-V[244], B97M-V [211], and ωB97M-V[247] functionals are specially fitted variants that use the non-local VV10 kernel to capture London-dispersion effects. They employ the non-self-consistent NL

correction by default and no additional NL keywords have to be used! A self-consistent treatment can be invoked by the `!SCNL` keyword.

Some notes on the NL corrections in ORCA

- Self-consistent calculations are not possible with the Hartree-Fock method.
- Analytic gradients are available, thus geometry optimizations with numerical frequencies can be computed.
- TD-DFT calculations are not yet available.
- Any calculations that require second derivatives of the NL functional are not yet possible. These are needed for real type perturbations in the CP-SCF solutions, e.g. for analytic Hessians, dipole polarizabilities, and double-hybrid gradients.
- Strictly imaginary perturbations such as NMR shielding and EPR g-tensors (both also with GIAOs), and hyperfine couplings are available.

Custom parameters

For NL, the parameters b, C, and a global scaling parameter of the NL correlation energy term can be adjusted via the `%method` block. The global scaling parameter `NLScal` should be used to scale the NL correction for methods that already partly cover long-range correlatione like double-hybrids.

```
!B3LYP NL

%method
  NLb    5.0
  NLC    0.0083
  NLScal 1.0
end
```

Note

In order to improve the scaling for larger systems, a distance cutoff was also introduced, controlled by the `vdWdistTCUT` flag in the `%method` block. The default value is 10 Å, so two grid points more than 10 Å away from each other do not correlate via the VV10 potential. This is already very conservative and has practically zero effect on the final energy, but can be altered if needed.

Example

In the following, we compute the energy of the argon dimer at a distance of 3.76 Å with the def2-TZVP basis set and using the B3LYP hybrid functional as an example. Here, we choose the non-self-consistent variant of the DFT-NL dispersion correction.

```
!B3LYP NL def2-TZVP

*xyz 0 1
  Ar  0.0 0.0 0.0
  Ar  0.0 0.0 3.76
*
```

The DFT-NL output for this example is shown below:

```
-------------------------------------------------------------------------------
                    post-SCF DFT-NL dispersion correction
-------------------------------------------------------------------------------

SCF Energy:    -1054.960511496
NL    Energy:      0.209449910
SC+NL Energy:  -1054.751061586
NL done in  :              0.7 sec
-------------------------------------------------------------------------------

[...]

----------------
TOTAL SCF ENERGY
----------------

[...]

DFT components:
N(Alpha)           :       17.999996328440 electrons
N(Beta)            :       17.999996328440 electrons
N(Total)           :       35.999992656880 electrons
E(X)               :      -47.880920358714 Eh
E(C)               :       -1.761923687962 Eh
NL Energy, E(C,NL) :        0.209449910227 Eh
E(XC)              :      -49.433394136450 Eh
```

Here, we find the B3LYP total energy (`SCF Energy`), the non-local contribution (`NL Energy`), and their sum (`SC+NL Energy`), which is the final total energy. In the `DFT components` section, the non-local contribution is listed separately (`NL Energy, E(C,NL)`) in order to be consistent with the `!SCNL` output.

Keywords

Table 3.19: Simple input keywords for the NL correction.

Keyword	Description
`NL`	Activates non-self-consistent NL correction
`SCNL`	Activates self-consistent NL correction

Table 3.20: `%method` block input keywords for the NL correction.

Keyword	Options	Description
`NLb`	`<real>`	Controls b parameter for NL
`NLC`	`<real>`	Controls C parameter for NL
`NLScal`	`<real>`	Controls the global scaling parameter for NL
`vdWdistTCUT`	`<real>`	Controls the correlation distance cutoff for NL

3.5 Semiempirical Methods

Semi-empirical quantum mechanical methods (SQM) are very useful to deal with many of very large systems. They use tailored approximations to reduce the computational cost drastically while keeping a reasonable level of accuracy for large parts of the chemical space. They are specifically useful to simulate raction paths with *NEB-TS* or to define a SQM region in *multi-scale approaches* like ONIOM.

3.5.1 Neglect of Differential Overlap: CNDO, INDO, NDDO

A number of methods based on the "neglect of differential overlap" [291, 292] are currently implemented for energies and analytic gradients inluding AM1[293], PM3[294], MNDO[295, 296, 297] and other variants of CNDO, INDO, and NDDO.The methods MNDO , AM1, and PM3 are available for main group elements only and arise from the work of the Dewar group. They have been optimized to reproduce molecular structure and energetics. The older CNDO/1,2 and INDO/1,2 were developed by the Pople group [298, 299, 300, 301, 302, 303, 304, 305, 306] and were designed to roughly mimic minimal basis *ab initio* calculations. The methods of the Zerner group (ZINDO/1,2 and ZINDO/S) are closely related to the older methods but have been well parameterized for transition metals too [307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317]. ZINDO/1 (and less so ZINDO/2) are suitable for geometry optimization. ZINDO/S gives good results for electronically excited states at moderate configuration interaction levels and is also successful for the calculation of electron and spin distributions in large transition metal complexes [312, 313, 314, 315, 316, 317]. The ZNDDO/1,2 methods have been implemented into ORCA as straightforward extensions of the corresponding INDO methods without changing any parameter. However, the methods benefit from the somewhat more accurate representation of the Coulomb interaction within the NDDO approximation [318, 319]. The preliminary experience with these methods is that they are better than the corresponding INDO methods for calculation of transition metal complex structures but on the whole have also similar deficiencies. The analytic gradients are available for all of these methods and can be used to produce reasonable molecular structures at low computational cost or to get preliminary insight in the behavior of the system under investigation. However, do not try to use ZINDO/S (or CNDO/S) for structure optimizations - it does not make sense and will lead to disastrous results because there is no accurate representation of nuclear repulsion in these methods.

Tip

Even though ORCA has no implementation of newer MNDO methods like PM6 or RM1, they can be used via ORCA's *External Optimizer feature* and MOPAC. This way, MOPAC can be interfaced to provide ORCA with energies and gradients via the respective wrapper script that can be downloaded from the ORCA external tools GitHub repository.

Basic Usage

The NDDO-based MNDO, AM1, and PM3 methods can be invoked via the simple input keywords `!MNDO`, `!AM1` and `!PM3`.

```
!MNDO
```

```
!AM1
```

```
!PM3
```

These and other methods can be further defined via the `%method` block, e.g. INDO with the INDO/1 parameter set as:

```
%method
   METHOD   INDO
   VERSION  INDO_1
end
```

Or by directly invoking the respective parameterization as `METHOD`

```
%method
   METHOD   INDO_1
end
```

> **Note**
>
> If you want you can also combine semiempirical methods with MP2 (energies only). For example use `Method AM1` and `DoMP2 true`. It is questionable if this makes the results of semiempirical calculations any better but at least it is possible in ORCA.

Custom Parameters

You can change the built-in semiempirical parameters in a straightforward fashion via the `%ndoparas` block. To do so, any element sepcific parameters can be defined via an array "P":

```
%ndoparas
   P[<atomic number>,<parameter number>] <real>
end
```

For example, we can set the resonance integral parameter BS (s shell beta) with the number 25 for Carbon (Z = 6) to 20 eV:

```
%ndoparas
   P[6,25] 20
end
```

A full list of parameters that can be modified can be found below. Most of them will only be interesting for expert users. The most commonly modified parameters are the Beta's (number 25 through 28). Note that most programs require a negative number here. In ORCA the resonance integrals are defined in a way that makes the Beta's positive.

Table 3.21: List of parameters that can be set in the `%ndoparas` block.

Parameter	Number	Description
Core Integrals (in eV):		
US	0	
UP	1	
UD	2	
UF	3	
Basis Set Parameters (double-zeta):		
NSH	4	Number of shells for the element
NZS	5	Number of Slater type orbitals for the s shell
ZS1	6	First exponent
ZS2	7	Second exponent
CS1	8	First contraction coefficient
CS2	9	Second contraction coefficient
NZP	10	Number of Slater type orbitals for the p shell
ZP1	11	
ZP2	12	
CP1	13	
CP2	14	
NZD	15	Number of Slater type orbitals for the d shell
ZD1	16	
ZD2	17	
CD1	18	
CD2	19	
NZF	20	Number of Slater type orbitals for the f shell
ZF1	21	
ZF2	22	
CF1	23	
CF2	24	

continues on next page

Table 3.21 – continued from previous page

Parameter	Number	Description
Resonance integral parameters (in eV):		
BS	25	s shell beta
BP	26	p shell beta
BD	27	d shell beta
BF	28	f shell beta
Number of electrons in the ground state:		
NEL	29	Total number of electrons (integer)
NS	30	Fractional occupation number of the s shell
NP	31	Fractional occupation number of the p shell
ND	32	Fractional occupation number of the d shell
NF	33	Fractional occupation number of the f shell
One center repulsion (gamma) integrals (in eV):		
GSS	34	
GSP	35	
GSD	36	
GSF	37	
GPP	38	
GPD	39	
GPF	40	
GDD	41	
GDF	42	
GFF	43	
Slater Condon parameters (in eV):		
F2PP	44	
F2PD	45	
F2DD	46	
F4DD	47	
G1SP	48	
G1PD	49	
G2SD	50	
G3PD	51	
R1SPPD	52	
R2SDPP	53	
R2SDDD	54	
The nuclear repulsion parameters for Dewar type models:		
NR1	55	
NR2	56	
NR3	57	
NR4	58	
NR5	59	
NR6	60	
NR7	61	
NR8	62	
NR9	63	
NR10	64	
NR11	65	
NR12	66	
NR13	67	
Nuclear attraction/repulsion parameter for MNDO/d:		
RHO	68	
Spin orbit coupling parameters:		
SOCP	69	SOC for the p shell
SOCD	70	SOC for the d shell
SOCF	71	SOC for the f shell

The "molecular" parameters are also set in the `%ndoparas` block using "`INTFA`" ("interaction factors");

```
%ndoparas
   INTFA[PP_PI] 0.585
  # The interaction factors exist for
  #  ss_sigma
  #  sp_sigma
  #  sd_sigma
  #  pp_sigma
  #  pd_sigma
  #  dd_sigma
  #  pp_pi
```

(continues on next page)

(continued from previous page)

```
  #  pd_pi
  #  dd_pi
  #  dd_delta
  # the parameter entering the Coulomb integrals
  # in INDO/S
   FGAMMA 1.2
end
```

Keywords

Table 3.22: Simple input keywords for the NDO-based methods.

Keyword	Description
AM1	Activates AM1
PM3	Activates PM3
MNDO	Activates MNDO
ZINDO/1	Activates ZINDO/1
ZINDO/2	Activates ZINDO/2
ZINDO/S	Activates ZINDO/S
ZNDDO/1	Activates ZNDDO/1
ZNDDO/2	Activates ZNDDO/2

Table 3.23: `%method` block input keywords for the NDO-based methods.

Keyword	Options	Description
METHOD	AM1	Activates AM1
	PM3	Activates PM3
	MNDO	Activates MNDO
	CNDO_1	Activates CNDO/1
	CNDO_2	Activates CNDO/2
	CNDO_S	Activates CNDO/S
	INDO_1	Activates INDO/1
	INDO_2	Activates INDO/2
	INDO_S	Activates INDO/S
	ZINDO_1	Activates ZINDO/1
	ZINDO_2	Activates ZINDO/2
	ZINDO_S	Activates ZINDO/S
	ZNDDO_1	Activates ZNDDO/1
	ZNDDO_2	Activates ZNDDO/2
	CNDO	Activates CNDO, requires a VERSION to be set
	INDO	Activates INDO, requires a VERSION to be set
	NDDO	Activates NDDO, requires a VERSION to be set
VERSION	CNDO_1	Sets the parameter set for CNDO to CNDO_1
	CNDO_2	Sets the parameter set for CNDO to CNDO_2
	CNDO_S	Sets the parameter set for CNDO to CNDO_S
	INDO_1	Sets the parameter set for INDO to INDO_1
	INDO_2	Sets the parameter set for INDO to INDO_2
	INDO_S	Sets the parameter set for INDO to INDO_S
	ZINDO_1	Sets the parameter set for INDO to ZINDO_1
	ZINDO_2	Sets the parameter set for INDO to ZINDO_2
	ZINDO_S	Sets the parameter set for INDO to ZINDO_S
	ZNDDO_1	Sets the parameter set for NDDO to ZNDDO_1
	ZNDDO_2	Sets the parameter set for NDDO to ZNDDO_2
	MNDO	Sets the parameter set for NDDO to MNDO
	AM1	Sets the parameter set for NDDO to AM1
	PM3	Sets the parameter set for NDDO to PM3

3.5.2 Extended Tight-Binding: GFN0-xTB, GFN-xTB, GFN2-xTB

ORCA is interfaced to the xtb program by Grimme and coworkers, allowing the user to request all kinds of calculations using the popular GFN0-xTB, GFN-xTB and GFN2-xTB Hamiltonians.[161, 320, 321] From the technical side, the user has to provide the executable provided by the Grimme group. The xtb program package can be obtained free of charge from https://github.com/grimme-lab/xtb/releases and detailed information on the usage of the `xtb` standalone program and its utilities can be found at https://xtb-docs.readthedocs.io/en/latest/contents.html. Only the file `bin/xtb` is used by ORCA . The user should copy this file into the directory where the other ORCA executables are located, and rename it as `otool_xtb`.

Warning

Please use the 6.7.1 version (or any later version) of xtb; older versions are not fully compatible with ORCA anymore or are missing features, for example it may not be possible to invoke the solvation model! Additionally, Windows users should copy `libiomp5md.dll` from the XTB directory to the ORCA directory.

The xTB method can be invoked via the `GFN0-xTB`, `GFN-xTB`, and `GFN2-xTB` keywords. The GFN-FF generic force-field can be invoked via `GFN-FF`:

```
! GFN0-xTB  # Synonym: XTB0
```

```
! GFN-xTB  # Synonym: XTB1
```

```
! GFN2-xTB  # Synonym: XTB2
```

```
! GFN-FF # Synonym: XTBFF
```

The following methods can be used in conjunction with XTB:

- Single Point Energy
- Energy and Gradient
- Optimization, using all kinds of constraints, relaxed surface scans, etc.
- Nudged-Elastic Band calculations
- Numerical Frequency Calculations
- Intrinsic Reaction Coordinate
- Molecular Dynamics Calculations
- QM/MM calculations

Note

- XTB0 is a non-self-consistent tight-binding method, and as such, its accuracy is generally inferior to XTB1 and XTB2 (and sometimes even XTBFF), despite that it is a few times faster than XTB1 and XTB2. From our experience, we only recommend XTB0 when both XTB1 and XTB2 exhibit qualitative failures for the system of interest.
- Please note that XTB0, XTB1 and XTB2 can also be used for the initial path generation or for the calculation of an initial TS structure on XTB level, both as input for the subsequent NEB calculation on a higher level of theory. For more details, please consult section *Nudged Elastic Band Method*.

Solvation in xTB

Three implicit solvation models can be requested in xTB calculations:

1. the analytical linearized Poisson-Boltzmann (ALPB)[322] solvation model
2. the domain decomposition COSMO (ddCOSMO)[323]
3. the extended conductor-like polarizable continuum model (CPCM-X).[149]

These models can be requested via the simple keyword input:

```
! ALPB(solvent) # use ALPB
```

```
! DDCOSMO(solvent) # use ddCOSMO
```

```
! CPCMX(solvent) # use CPCMX
```

where `solvent` is any of the solvents in Table 3.24.

They can further be envoked via the `%xtb` block, here exemplarily shown for ALPB:

```
%xtb
  DOALPB true
  ALPBSOLVENT "water"
end
```

Table 3.24: List of available solvents for the different implicit solvation methods that can be used with `xtb`.

Solvent	ALPB	ddCOSMO	CPCM-X
1,2,4-trimethylbenzene			X
1,2-dichloroethane			X
1,4-dioxane / dioxane	X	X	
1-butanol / butanol			X
1-chlorohexane / chlorohexane			X
1-decanol / decanol			X
1-fluorooctane			X
1-heptanol / heptanol			X
1-hexanol / hexanol			X
1-iodohexadecane / hexadecyliodide			X
1-nonanol / nonanol			X
1-octanol / octanol	X	X	X
1-pentanol / pentanol			X
1-propanol / propanol			X
2,2,4-trimethylpentane / isooctane			X
2,6-dimethylpyridine			X
2-butanol / secbutanol			X
2-methoxyethanol / methoxyethanol			X
2-methyl-1-propanol / isobutanol			X
2-methylpyridine / 2methylpyridine			X
2-propanol / isopropanol			X
4-methyl-2-pentanone / 4methyl2pentanone			X
acetic acid / aceticacid			X
acetone	X	X	
acetonitrile / mecn / ch3cn	X	X	X
acetophenone			X
aniline	X	X	X
anisole			X
benzaldehyde	X	X	
benzene	X	X	X
benzonitrile			X
benzyl alcohol / benzylalcohol			X
bromobenzene			X
bromoethane			X
bromoform			X
butanone			X
butyl ethanoate / butyl acetate / butylacetate			X
n-butylbenzene / butylbenzene			X

continues on next page

Table 3.24 – continued from previous page

Solvent	ALPB	ddCOSMO	CPCM-X
sec-butylbenzene / secbutylbenzene			X
tert-butylbenzene / tbutylbenzene			X
carbon disulfide / carbondisulfide / cs2	X	X	X
carbon tetrachloride / ccl4			X
chlorobenzene			X
chloroform / chcl3	X	X	X
conductor		X	
m-cresol / mcresol			X
cyclohexane			X
cyclohexanone			X
decalin			X
n-decane / decane			X
dibromomethane			X
dibutylether			X
o-dichlorobenzene / odichlorobenzene			X
dichloromethane / ch2cl2 / dcm	X	X	X
diethyl ether / diethylether	X	X	X
diisopropyl ether / diisopropylether			X
n,n-dimethylacetamide / dimethylacetamide			X
n,n-dimethylformamide / dimethylformamide / dmf	X	X	X
dimethylsulfoxide / dmso	X	X	X
diphenylether			X
n-dodecane / dodecane			X
ethanol	X	X	X
ethyl acetate / ethylacetate / ethanoate	X	X	X
ethyl phenyl ether / ethoxybenzene			X
ethylbenzene			X
fluorobenzene			X
furan / furane	X	X	
n-heptane / heptane			X
n-hexadecane / hexadecane	X	X	X
n-hexane / hexane	X	X	X
iodobenzene			X
isopropylbenzene			X
p-isopropyltoluene / isopropyltoluene			X
mesitylene			X
methanol	X	X	X
n-methylformamide / methylformamide			X
nitrobenzene / phno2			X
nitroethane			X
nitromethane / meno2	X	X	X
o-nitrotoluene / onitrotoluene			X
n-nonane / nonane			X
n-octane / octane			X
n-pentadecane / pentadecane			X
octanol(wet) / wetoctanol / woctanol	X	X	
n-pentane / pentane			X
perfluorobenzene / hexafluorobenzene			X
phenol	X	X	
pyridine			X
tetrachloroethene / c2cl4			X
tetrahydrofuran / thf	X	X	X
tetrahydrothiophene-s,s-dioxide / tetrahydrothiophenedioxide / sulfolane			X
tetralin			X
toluene	X	X	X
tributylphosphate			X
triethylamine			X
n-undecane / undecane			X
water / h2o	X	X	X
xylene			X

Note

- If jobs are run over several nodes, the number of cores used by the XTB tool might be lower than requested via the pal keyword.

Keywords

Table 3.25: Simple input keywords for the extended tight-binding methods.

Keyword	Description
XTB	Activates GFN2-xTB
XTB0	Activates GFN0-xTB
GFN0-XTB	Activates GFN0-xTB
XTB1	Activates GFN1-xTB
GFN1-XTB	Activates GFN1-xTB
XTB2	Activates GFN2-xTB
GFN2-XTB	Activates GFN2-xTB
XTBFF	Activates GFN-FF
GFN-FF	Activates GFN-FF
ALPB(<solvent>)	Activates ALPB solvation for a <solvent>, e.g. ALPB(Water)
DDCOSMO(<solvent>)	Activates DDCOSMO solvation for a <solvent>, e.g. DDCOSMO(Water)
CPCMX(<solvent>)	Activates CPCMX solvation for a <solvent>, e.g. CPCMX(Water)

Table 3.26: %xtb block input keywords for the extended tight-binding methods.

Keyword	Options	Description
XTBINPUTSTRING	"<string>"	Parses string to the otool_xtb call
XTBINPUTSTRING2	"<string>"	Parses string to the otool_xtb call
NPROCS	<integer>	Controls number of processors for the otool_xtb run, default is ORCA's PAL setting
MAXCORE	<integer>	Memory in MB reserved for the xtb run, default is ORCA's maxcore setting
ACCURACY	<real>	Accuracy setting for --acc in otool_xtb, default is ORCA's accuracy x 1.e6
DOALPB	true/false	Activates ALPB solvation in xtb run
ALPBSOLVENT	"<solvent>"	Controls the <solvent> used for ALPB
DODDCOSMO	true/false	Activates DDCOSMO solvation in xtb run
DDCOSMOSOLVENT	"<solvent>"	Controls the <solvent> used for DDCOSMO
EPSILON	<real>	Controls the dielectric constant, only for ddCOSMO
DOCPCMX	true/false	Activates CPCMX solvation in xtb run
CPCMXSOLVENT	"<solvent>"	Controls the <solvent> used in CPCMX
ETEMP	<real>	Controls electronic temperature --etemp in otool_xtb

3.5.3 Native GFN-xTB and GFN2-xTB

Added in version 6.1.0: Native implementation of GFN1-xTB and GFN2-xTB

ORCA features its own native implementation of GFN-xTB and GFN2-xTB. Unlike the interface to the xtb program, the native xTB implementation can use the *SCF* and *plotting* infrastructure.

The native xTB methods can be invoked via the !Native-GFN-xTB, !Native-GFN2-xTB simple keywords:

```
! Native-GFN-xTB  # Synonym: Native-GFN1-xTB, Native-XTB1
```

```
! Native-GFN2-xTB # Synonym: Native-XTB2
```

Warning

For a faster SCF restart based on atomic charges and multipole moments, the native xTB methods write a basename.xtbw file. If such an .xtbw file is present, it will be used for the restart instead of the .gbw file.

Custom parameterization

The native implementations feature the use of custom semi-empirical parameters by reading a JSON parameter file with the name `basename.xtb.json`. This file has the same format as the JSON file in the dxtb program (dxtb documentation), and can be generated by writing the internal parameters. Reading and writing of semi-empirical parameters can be turned on and off via the `%method` block:

```
%method
  XTBParamFile "orca.xtb.json" # set parameter filename to orca.xtb.json
  WriteXTBParam true      # if true: write parameters to orca.xtb.json file
  WriteXTBParamDry false  # if true: write parameters to orca.xtb.json file and␣
↪exit
  ReadXTBParam false      # if true: read parameters from orca.xtb.json file
end
```

By default the parameter file will be named according to the basename of your calculation (`basename.xtb.json`), but other names can be chosen with the `XTBParamFile` option in the method block. For a dry run that just generates the parameter file from default parameters set `WriteXTBParamDry true`.

> **Warning**
>
> Please note that the native xTB implementation does not allow changes in the basis.

Spin-polarized native xTB

The energy expressions of the original GFNn-xTB methods are spin-independent. Therefore, calculations are typically performed using RHF/RKS-type wavefunctions, even for open-shell systems. Since spin does not enter the energy expression, electronic states with higher spin multiplicity will always be higher in energy compared to the respective low-spin states. This shortcoming is addressed by the spin-polarized xTB methods. The spin-polarized variants [324], termed spGFN1-xTB and spGFN2-xTB, employ a simple on-site spin-dependent term, which can stabilize high-spin states and allows for screening of potential high-spin configurations using xTB methods.

The native spin-polarized xTB methods can be invoked via the `!Native-spGFN1-xTB`, `!Native-spGFN2-xTB` simple keywords:

```
! Native-spGFN1-xTB  # Synonym: Native-spGFN-xTB, Native-spXTB1
```

```
! Native-spGFN2-xTB # Synonym: Native-spXTB2
```

Note that the spin-polarization term will only affect high-spin states. The computational cost increases compared to RHF/RKS-type calculations, as the underlying UHF/UKS-type wavefunctions require two Fock matrix diagonalizations instead of one—approximately doubling the cost.

SCF with native xTB

The native xTB methods, by default, employ special SCF settings that mimic the SCF procedure of the `xtb` program and override all other ORCA SCF settings. These special settings take advantage of xTB-specific features, most notably, that the Fock matrix can be constructed from atomic charges and multipole moments alone. Due to these shortcuts, the special SCF settings are typically faster for xTB methods than the default ORCA SCF. However, the default SCF is more robust and may succeed in converging problematic cases where the `xtb` SCF fails.

To use the standard ORCA SCF infrastructure instead, set `UseXTBMixer` to false in the `%scf` block:

```
%scf
  UseXTBMixer false    # if true (default): special xTB SCF is used
end
```

With the ORCA default SCF infrastructure, challenging electronic structures can often be converged, but the results should be used with caution. In cases where SCF convergence has to be forced, the GFNn-xTB Hamiltonians are often not adequate to describe the system at hand. In such cases, Mulliken charges should be checked for sanity, and it might be helpful to slightly adjust the element-wise pair parameters via the parameter file described above.

Warning

Note that all SCF iterations are direct with xTB methods.

FOD with native xTB

To better account for static correlation, the xTB methods use *finite temperature SCF* by default, with an electronic temperature of 300 K. The corresponding *fractional occupation number weighted density (FOD)* is not printed by default, since 300 K is a too low temperature to use it as a diagnostic tool. The FOD printout can be enabled by setting `XTBFOD` to true in the `%scf` input block. The electronic temperature can be controlled–just as for other methods–via the `SmearTemp` keyword:

```
%scf
  XTBFOD true    # if true: Enable FOD printout and add FOD to density container
  SmearTemp 5000 # 5000 K is typically used for FOD plots with xTB methods
end
```

Solvation in native xTB

The native implementation of xTB includes the ALPB solvation model with the same solvents available as in the `xtb` program (*see above*). It is controlled with the same keywords as the external ALPB via the simple keyword input:

```
! ALPB(solvent) # use ALPB
```

In addition, the native implementations can be used with other *solvation models* available in ORCA, but one should consider that the CPCM and SMD models are quite expensive compared to an xTB calculation, and therefore the faster ALPB solvation model is recommended.

Keywords

Table 3.27: Simple input keywords for the native extended tight-binding methods.

Keyword	Description
`NATIVE-XTB`	Activates native GFN2-xTB
`NATIVE-spXTB`	Activates native spGFN2-xTB
`NATIVE-XTB1`	Activates native GFN1-xTB
`NATIVE-spXTB1`	Activates native spGFN1-xTB
`NATIVE-GFN-XTB`	Activates native GFN1-xTB
`NATIVE-spGFN-XTB`	Activates native spGFN1-xTB
`NATIVE-GFN1-XTB`	Activates native GFN1-xTB
`NATIVE-spGFN1-XTB`	Activates native spGFN1-xTB
`NATIVE-XTB2`	Activates native GFN2-xTB
`NATIVE-spXTB2`	Activates native spGFN2-xTB
`NATIVE-GFN2-XTB`	Activates native GFN2-xTB
`NATIVE-spGFN2-XTB`	Activates native spGFN2-xTB
`ALPB(<solvent>)`	Activates ALPB solvation for a `<solvent>`, e.g. ALPB(Water)

Table 3.28: `%method` block input keywords for the native extended tight-binding methods.

Keyword	Options	Description
WRITEXTBPARAM	true/false	Write semi-empirical parameters to xtb.json file
WRITEXTBPARAMDRY	true/false	Write semi-empirical parameters to xtb.json file and abort
READXTBPARAM	true/false	Read semi-empirical parameters from xtb.json file
XTBPARAMFILE	"filename.xtb.json"	Specifies xtb.json filename to "filename.xtb.json"

Table 3.29: %scf block input keywords for the native extended tight-binding methods.

Keyword	Options	Description
XTBFOD	true/false	Enable FOD printout
UseXTBMixer	true/false	Use special SCF settings similar to the ones in xtb

3.6 Composite Methods (3c methods)

Composite *HF* and *DFT* methods utilize relatively small tailored *basis sets* and special corrections to achieve high accuracies at a fraction of the computational cost of a calculation approaching the basis set limit. The most prominent composite methods are the so-called "3c" methods by Grimme and co-workers.

3.6.1 HF-3c

HF-3c is a fast *Hartree-Fock* based method developed for computation of structures, vibrational frequencies and non-covalent interaction energies in huge molecular systems [325]. The starting point for evaluating the electronic energy is a standard HF calculation with a small Gaussian AO basis set. The used so-called MINIX basis set consists of different sets of basis functions for different groups of atoms as shown in table Table 3.30. In ORCA, HF-3c is available for all elements up to Pu (Z = 94).

Table 3.30: Composition of the MINIX basis set.

element	basis
H-He, B-Ne	MINIS
Li-Be	MINIS+1(p)
Na-Mg	MINIS+1(p)
Al-Ar	MINIS+1(d)
K-Zn	SV
Ga-Kr	SVP
Rb-Rn	def2-SVP with def-ECPs
Fr-Lr	def-SVP with def-ECPs

Three terms are added to correct the HF energy $E_{\text{tot}}^{\text{HF/MINIX}}$ in order to include London dispersion interactions, to account for the BSSE and to correct for basis set deficiencies, i.e. overestimated bond lengths. The corrected total energy is therefore calculated as

$$E_{\text{tot}}^{\text{HF-3c}} = E_{\text{tot}}^{\text{HF/MINIX}} + E_{\text{disp}}^{\text{D3(BJ)}} + E_{\text{BSSE}}^{\text{gCP}} + E_{\text{SRB}}. \tag{3.23}$$

The first correction term $E_{\text{disp}}^{\text{D3(BJ)}}$ is the atom-pair wise London dispersion energy from the *D3 dispersion correction* scheme[272] applying Becke-Johnson (BJ) damping [277, 278, 279] (see section *Grimme's DFT-D3 and DFT-D4*). The second term $E_{\text{BSSE}}^{\text{gCP}}$ denotes the *geometrical counterpoise (gCP) correction* [124] to treat the BSSE (see section *Geometrical Counterpoise Correction (gCP)*). For the HF-3c method, the three usual D3 parameters s_8, a_1 and a_2 were re-fitted using reference interaction energies of the complexes of the S66 test set [285]. This results in $s_8 = 0.8777$, $a_1 = 0.4171$ and $a_2 = 2.9149$. The parameter s_6 was set to unity as usual to enforce the correct asymptotic limit and the gCP correction was already applied in this fitting step.

The last term E_{SRB} is a short-ranged correction to deal with basis set deficiencies which occur when using small or minimal basis sets. It corrects for systematically overestimated covalent bond lengths for electronegative elements

and is calculated as a sum over all atom pairs:

$$E_{\text{SRB}} = -s \sum_{A}^{\text{Atoms}} \sum_{B \neq A}^{\text{Atoms}} (Z_A Z_B)^{3/2} \exp(-\gamma (R_{AB}^{0,\text{D3}})^{3/4} R_{AB})$$

Here, $R_{AB}^{0,\text{D3}}$ are the default cut-off radii as determined *ab initio* for the D3 scheme [272] and Z_A, Z_B are the nuclear charges. This correction is applied for all elements up to argon. The empirical fitting parameters $s = 0.03$ and $\gamma = 0.7$ were determined to produce vanishing HF-3c total atomic forces for B3LYP-D3(BJ)/def2-TZVPP equilibrium structures of 107 small organic molecules. More details can be found in the original publication [325].

The HF-3c method can only be invoked with a simple keyword:

```
! HF-3c
```

`! HF-3c` is a compound keyword and equals `! HF MINIX D3BJ GCP(HF/MINIX) PATOM`, hence no basis set etc. has to be specified. The `PATOM` guess is selected since the grid construction for the default guess can take more time than an actual SCF step. The guess can only be overwritten manually in the %method section.

The default mode for the integral handling is set to `Conventional`. The storing of the two-electron integrals on disk or in memory if possible leads to large computational savings. In case one want to use the `Direct` mode, this has to be specified in the %scf input section:

```
%scf
  SCFmode Direct
end
```

The output gives the used parameters and the correction itself for D3 and gCP separately. As the SRB correction is also calculated with the otool_gcp, the results are given in the gCP output section. The `Total correction to HF/MINIX` is the sum of all three corrections (D3, gCP and SRB) and `FINAL SINGLE POINT ENERGY` is the total HF-3c energy as given in equation (3.23).

```
-----------------------------------------------------------------------------
                          DFT DISPERSION CORRECTION

                                DFTD3 V2.1  Rev 6
                          USING Becke-Johnson damping
-----------------------------------------------------------------------------
The default Hartree-Fock  is recognized
Active option DFTDOPT                   ...         4

molecular C6(AA) [au] = 1689.256597


            DFT-D V3
 parameters
 using HF/MINIX parameters
 s6 scaling factor         :     1.0000
 a1 scaling factor         :     0.4171
 s8 scaling factor         :     0.8777
 a2 scaling factor         :     2.9149
 ad hoc parameters k1-k3   :    16.0000     1.3333    -4.0000

 Edisp/kcal,au: -32.163184627631  -0.051255291794
 E6   /kcal   : -18.007221978
 E8   /kcal   : -14.155962649
 % E8         :  44.012938437

-------------------------   ----------------
Dispersion correction           -0.051255292
-------------------------   ----------------
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------------

                  g C P  - geometrical counterpoise correction

-----------------------------------------------------------------------------
Method:  hf/minix

Parameters:   sigma        eta         alpha        beta
              0.1290      1.1526      1.1549       1.1763
Egcp:       0.0723150636 a.u.
Ebas:      -0.0636976872 a.u.
------------------   -----------------
gCP+bas correction         0.008617376
------------------   -----------------
----------------------------   ----------------
Total correction to HF/MINIX       -0.042637915
----------------------------   ----------------


-------------------------   --------------------
FINAL SINGLE POINT ENERGY      -163.002895262171
-------------------------   --------------------
```

For the elements up to Xe, the default initial guess is a Hückel guess. Beyond Xe, the guess mode is changed to `HCORE` since no Hückel parameters for the respective ECP bases are available and other models are not implemented at the moment. For calculations with only lighter elements and therefore no ECPs, the ECP printouts in the output file can be ignored.

3.6.2 B97-3c

B97-3c[326] is another composite DFT method designed for thermochemistry, structures, and noncovalent interactions specifically also for transition metal chemistry and other stronger correlated systems. It is based on the B97 GGA including the *D3(BJ) dispersion correction* with three-body contribution, a short range bond length correction, and a modified, stripped-down triple-ζ basis termed def2-mTZVP, the computational cost of this method is between that of HF-3c and *PBEh-3c* (for large systems roughly two times more expensive than *HF-3c*). In ORCA, B97-3c is available for all elements up to Pu (Z = 94).

B97-3c can be invoked via simple input keyword:

```
! B97-3c
```

3.6.3 r^2SCAN-3c

The r^2SCAN-3c composite method[221] is available as robust "Swiss army knife" electronic structure method for thermochemistry, geometries and non-covalent interactions and has shown in preliminary tests consistent performance for both open and closed shell transition metal complexes. It is based on the r^2SCAN[216] meta-GGA combined with the *D4 dispersion correction*[327] and the *geometrical counter poise-correction*[124]. The modified triple-ζ basis set, def2-mTZVPP, is larger and more consistent for the light main-group elements and almost as computationally efficient as the def2-mTZVP basis set of *B97-3c*. The computational cost of r^2SCAN-3c is slightly larger than *B97-3c*. In ORCA, r^2SCAN-3c is available for all elements up to Lr (Z = 103). It is invoked with the simple keyword

```
! r2SCAN-3c
```

3.6.4 PBEh-3c

PBEh-3c is a highly efficient electronic structure approach performing particularly well in the optimization of geometries and for interaction energies of non-covalent complexes.[239] Here, a global hybrid variant of the Perdew-Burke-Ernzerhof (PBE) functional with a relatively large amount of non-local Fock-exchange (42%) is employed with a valence-double-zeta Gaussian AO basis set (def2-mSVP). Basis set superposition errors (BSSE) and London dispersion effects are accounted for by the *gCP* and *D3* schemes, respectively (see above). The basis set is constructed such that:

Table 3.31: Composition of the def2-mSVP basis set.

element	basis
H	def2-SV(P) (α scaled by 1.2)
He	def2-SVP(-p)
Li-Be,Na-Kr	def2-SV(P)
B,Ne	Ahlrichs' DZ + Polarization from def2-SVP
C-F	Ahlrichs' DZ + Polarization from 6-31G*
Rb-Rn	def2-SVP with def2 ECPs
Fr-Lr	def-SVP with def-ECPs

For inter- and intramolecular BSSE the *gCP* expression from Eq. (2.25) is used but with a damping function (similar to the zero-damping in Eq. (3.14)). This damping improves the thermochemistry of the method significantly compared with the non-damped version. London dispersion effects are accounted for by the *DFT-D3* (BJ-damping) scheme including the three-body term. Compared to the related HF-3c approach, the PBEh-3c is somewhat more costly, however, it yields much better geometries. These are roughly of MP2-quality (or even better for non-covalent structures) but may be computed at much lower cost. Due to the moderate amount of non-local Fock exchange, the method is less prone to self-interaction errors (as in GGAs) but still applicable in cases when Hartree-Fock fails (strongly correlated systems). In ORCA, PBEh-3c is available for all elements up to Pu (Z = 94).

The PBEh-3c method may be invoked with the simple keyword:

```
! PBEh-3c
```

Identical to HF-3c, the default initial guess for all elements up to Xe is a Hückel guess. Beyond Xe, the guess mode is changed to `HCORE`. For calculations with only lighter elements and therefore no ECPs, the ECP printouts in the output file can be ignored.

3.6.5 B3LYP-3c

B3LYP-3c is a method combination introduced by Grimme and co-workers to efficiently calculate gas-phase infrared spectra.[240] It combines the standard B3LYP functional with the *D3(BJ)-ATM dispersion correction*, a def2-mSVP basis set, and a *geometrical counterpoise correction*. In ORCA, B3LYP-3c is available for all elements up to Pu (Z = 94). B3LYP-3c can be invoked via simple input keyword:

```
! B3LYP-3c
```

or it can be constructed manually from its components via:

```
!B3LYP D3BJ GCP(DFT/SV(P)) def2-mSVP ABC
```

3.6.6 ωB97X-3c

The ωB97X-3c composite method[49] is based on the ωB97X-V functional and combines a tailored and molecule-optimized polarized valence double-ζ (vDZP) basis set and a specifically adapted D4 dispersion correction. The vDZP basis set employs large-core ECPs and shows only very small basis set superposition and incompleteness errors compared to conventional double-ζ basis sets. In thorough tests on standard benchmarks sets, the ωB97X-3c method was shown to be on par with well-performing hybrid DFT methods in a quadruple-ζ basis set at a fraction of their computational cost. ωB97X-3c is consistently available for all elements up to Rn (Z = 1–86).

It is invoked with the simple keyword:

```
! wB97X-3c
```

The vDZP basis set alone is utilized as follows (note that the corresponding large-core ECPs are called automatically):

```
! vDZP
```

3.6.7 Keywords

Table 3.32: Simple input keywords for the 3c composite methods.

Keyword	Description
`HF-3C`	Invokes the *HF-3c* method
`B97-3C`	Invokes the *B97-3c* method
`R2SCAN-3C`	Invokes the *r^2SCAN-3c* method
`PBEH-3C`	Invokes the *PBEh-3c* method
`B3LYP-3C`	Invokes the *B3LYP-3c* method
`WB97X-3C`	Invokes the *ωB97X-3c* method

3.7 Analytic Density Functional Theory (ADFT)

The method that has come to be known as "analytic density functional theory" (ADFT) is a variant of DFT that avoids the very common use of numerical integration. Early approaches to a grid DFT were relying on diagonalizing the density matrix and applying the DFT functional to the eigenvalues of the density followed by back transformation. These methods work but have been abandoned in favor of numerical integration.

ORCA features an alternative approach, that is, however, still limited to the most elementary of all density functionals, namely the Hartree-Fock Slater (HFS) or, in a slightly parameterized form, the X-Alpha method. The method is based on fitting the Coulomb and exchange potentials to separate auxiliary basis sets. In the case of the Coulomb interaction, this is very well known in form of the RI-J approximation. For the exchange, the ADFT construction is based on a related but more tricky development given the nonlinear dependency of the Slater-exchange on the density.

The exchange energy is:

$$E_X^\sigma = -\frac{3}{4}\left(\frac{6}{\pi}\right)^{\frac{1}{3}} \int \rho_\sigma^{4/3}(\mathbf{r})$$

Leading to the exchange potential:

$$V_X^\sigma(\mathbf{r}) = -\left(\frac{6}{\pi}\right)^{\frac{1}{3}} \rho_\sigma^{1/3}(\mathbf{r})$$

$$C_X = -\left(\frac{6}{\pi}\right)^{\frac{1}{3}} = -1.24\ldots$$

Replacing C_X by $\frac{3}{2}\alpha C_X$ leads to the X-Alpha method. Here, α is an empirical parameter with default value 2/3 in order to recover the HFS method. Empirically, it has been found, however, that values between 2/3 and 1 are more suitable with the value 0.7 being a widely accepted default.

We now have to deal with the odd power law, using techniques developed by Cook and also by Dunlap and co-workers. The idea here is to expand $\rho_\sigma(r)^{\frac{1}{3}}$ itself in another auxiliary basis set:

$$\rho_\sigma\left(\mathbf{r}\right)^{1/3} = \sum_K a_K^\sigma A_K(\mathbf{r})$$

It turns out that we also need (or perhaps not need but want):

$$\rho_\sigma\left(\mathbf{r}\right)^{2/3} = \sum_K b_K^\sigma B_K(\mathbf{r})$$

The functions {A} and {B} are uncontracted exchange auxiliary basis functions. The easiest choice for these auxiliary basis sets is to derive them from the Coulomb fitting basis set, possibly with restricting the maximum angular momentum. Alternatively, the program can provide one of presently four fitting basis of increasing size called XFIT/1 through XFIT/4.

Based on theoretical arguments, the exponents of these fitting bases are scaled differently. The exponents of the A-basis are scaled by the factor 1/3 whereas those of the B-basis are scaled by 2/3. Dunlap remarked that it is sufficient to scale the exponents of the s-functions and we have followed that recommendation but have not observed large differences to scaling all exponents.

The determination of the fitting coefficients {a} and {b} leads to a set of nonlinear equations that can, for example, be solved by a Newton-Raphson procedure. They can be more efficiently also calculated by numerical integration, but that defeats the purpose of the methodology. The equations to be satisfied are:

$$\frac{\partial E}{\partial a_K^\sigma} = C_X \left\{ \sum_{\mu\nu} P_{\mu\nu}^\sigma (\mu\nu A_K) - \sum_{LM} a_L^\sigma b_M^\sigma (A_K A_L B_M) \right\}$$

$$\frac{\partial E}{\partial b_M^\sigma} = \frac{1}{2} C_X \left\{ -\sum_{KL} a_K^\sigma a_L^\sigma \left(A_K A_L B_M\right) + \sum_N b_N^\sigma (B_M B_N) \right\}$$

Where $(A_K A_L B_M)$ are 3-center overlap integrals, μ, ν are basis functions and P^σ is the density matrix for spin $\sigma = \alpha, \beta$. One can bring that into the form:

$$\left(\mathbf{H}(\mathbf{a})^{AB} \overline{\mathbf{S}}^{-1} \mathbf{H}(\mathbf{a})^{AB+}\right) \mathbf{a} = \mathbf{P}^X$$

With

$$P_K^{X;\sigma} = \sum_{\mu\nu} P_{\mu\nu}^\sigma (\mu\nu A_L)$$

$$\overline{S}_{MN} = (B_M B_N)$$

$$H_{KM}^{AB;\sigma} = \sum_L a_L^\sigma \left(A_K A_L B_M\right)$$

Which needs to be solved for a. We then obtain b from

$$\mathbf{b} = \overline{\mathbf{S}}^{-1} \mathbf{H}(\mathbf{a})^{AB+} \mathbf{a}$$

This is essentially the procedure followed in the ORCA implementation. Optimized code has been written for the three center overlap integrals.

The implementation in ORCA covers closed and spin-unrestricted energies, analytic geometry gradients, response properties and excitation energies as well as numerical Hessians. The efficiency of the implementation is such that energies are slower than with numeric integration while all other steps are executed faster.

One unique opportunity that ADFT offers is to scale the exchange for each atom in the molecule differently. This seems warranted, because X-Alpha studies on atoms revealed that optimal α values change between atoms. Some contemplation shows that this can be achieved by scaling the density matrix for the exchange calculation as follows:

$$P^{\sigma}_{\mu_A\nu_B} \rightarrow P^{\sigma}_{\mu_A\nu_B} \left(\frac{3}{2}\alpha_A\right)^{3/8} \left(\frac{3}{2}\alpha_B\right)^{3/8}$$

Alpha-values can be chosen in a variety of ways other than assigning a global value of 0.7. For example, the alpha values can be fitted to reproduce atomic Hartree-Fock energies or atomic coupled-cluster energies. They could also be optimized to reproduce molecular energetics.

The usage of ADFT in ORCA is simple:

```
! ADFT def2-SVP def2/J XFit/3
# or ADFT(XAlpha), ADFT(HF), ADFT(CC)
# default parameters for HF are H-Kr
# default parameters for CC are H-Zn
# XFIT/1 -- XFIT/4 exists but only /3 and /4 are recommended

%method ADFTNRTol 1e-9           # tol. for NR equation solution (def=1e-9)
        ADFTNRMaxIter 6          # max no of NR iterations (def=15)
        ADFTLmaxAuxJ  5          # max L allowed in the AUXJ basis (def=3)
        ADFTLmaxAuxX  5          # max L allowed in the AUXX basis (def=2)
        ADFTXAlpha  [36] 0.71337 # array with alpha values for each element
end
```

Note

- ADFT is really limited to the HFS/X-Alpha method. It makes no sense to input the name of any other functional. It will be ignored. This is due to the way the exchange is fitted which is very specific to this particular functional and the whole theory would be completely different for other functionals.
- The implementation is experimental. It is motivated by the intellectual beauty of the construction and the fact that in early computational chemistry, users got a lot of mileage out of the method. Hence, this is a "back to the roots" kind of movement with the idea to perhaps take grid free DFT to new places in the future.

Relevant Papers:

1. Werpetinski, Katrina S.; Cook, Michael. Grid-free density-functional technique with analytical energy gradients. *Phys. Rev. A*, **1995**, 52, R3397–R3400. DOI: 10.1103/PhysRevA.52.R3397.
2. Werpetinski, Katrina S.; Cook, Michael. A new grid-free density-functional technique: Application to the torsional energy surfaces of ethane, hydrazine, and hydrogen peroxide. *The Journal of Chemical Physics*, **1997**, 106 (17), 7124–7138. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/106/17/7124/19164467/7124_1_online.pdf, DOI: 10.1063/1.473734.
3. Zope, Rajendra R.; Dunlap, Brett I. Slater's Exchange Parameters α for Analytic and Variational Xα Calculations. *Journal of Chemical Theory and Computation*, **2005**, 1 (6), 1193–1200. PMID: 26631663. arXiv:https://doi.org/10.1021/ct050166w, DOI: 10.1021/ct050166w.

- Neese, F. 2025, in preparation

3.8 Random Phase Approximation (RPA)

The random phase approximation is a rather different DFT method that is based on the frequency dependent response function. It is more expensive than standard DFT (same order of magnitude as RI-MP2), has many attractive features, in particular it's good to excellent accuracy while being essentially non-empirical. The seminal work that established RPA in a quantum chemical context is due to Furche and is based on the original work of Langreth and Perdew. The implementation in ORCA closely follows the work of Görling and co-workers who we also gratefully acknowledge for help with the implementation.

The RPA energy is calculated using some set of input orbitals and orbitals energies (in practice, PBE orbitals and energies are recommended). The calculation proceeds by a numerical integration over frequencies:

$$E_{RPAC} \approx -\frac{\pi}{2} \sum_g w_g \sum_K h_K\left(\omega_g\right)$$

$$h_K(\omega) = -log\left(1 + x_K(\omega)\right) + x_K(\omega)$$

Where x_K are the negative eigenvalues of the Kohn-Sham response matrix expressed in an auxiliary fitting basis (an auxiliarly basis of the '/C' type).

$$X_{KL}(\omega) = \sum_i \left(\mathbf{X}^i\right)^T \mathbf{Y}^i(\omega)$$

With

$$X^i_{aK} = (ia|K)$$

$$Y^i_{aK}(\omega) = (ia|K)\,\lambda_{ia}(\omega)$$

$$\lambda_{ia} = f_\sigma \frac{\epsilon_i - \epsilon_a}{\left(\epsilon_i - \epsilon_a\right)^2 + (\omega)^2}$$

Here ϵ_i, ϵ_a are occupied and virtual orbital energies from the SCF calculation $(ia|K)$ is a three-index electron repulsion integral in the MO basis $f_\sigma = 4$ for closed-shell states and $f_\sigma = 2$ for spin-unrestricted calculations. The numerical integration is carried out with a Gauss-Legendre quadrature using the weights and frequencies given by:

$$w_g = w_{g0} \frac{5}{\left(1 - \omega_{g0}\right)^2}$$

$$\omega_g = \frac{5}{2}\left(\frac{1 + \omega_{g0}}{1 - \omega_{g0}}\right)$$

Where w_{g0} and ω_{g0} are the Gauss-Legendre weights and roots mapped on the interval -1 to 1.

The usage of RPA in ORCA is straightforward

```
! PBE RPAC def2-SVP def2/J def2-SVP/C
# we need a /C basis for RPAC and the /J basis for PBE

%method RPACNPoints 50  # Number of numerical integration points (def=50)
end
```

Note

- This is a pilot implementation that only features closed-shell and spin-unrestricted energies. Gradients are done numerically and density matrices are the ones from the underlying DFT calculation, NOT genuine RPA densities.
- Excited states and response properties are NOT available with this method.
- The method requires an integral transformation with scaling O(N^4) and on the order of 50 diagonalizations of the response matrix which is of dimension NAUXC. This can become expensive for larger molecules.

1. Langreth, D.C.; Perdew, J.P. The exchange-correlation energy of a metallic surface. *Solid State Communications*, **1975**, 17 (11), 1425–1429. DOI: 10.1016/0038-1098(75)90618-3.
2. Furche, Filipp. Molecular tests of the random phase approximation to the exchange-correlation energy functional. *Phys. Rev. B*, **2001**, 64, 195120. DOI: 10.1103/PhysRevB.64.195120.
3. Furche, Filipp. Developing the random phase approximation into a practical post-Kohn–Sham correlation model. *The Journal of Chemical Physics*, **2008**, 129 (11), 114105. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.2977789/13579843/114105_1_online.pdf, DOI: 10.1063/1.2977789.
4. Trushin, Egor; Thierbach, Adrian; Görling, Andreas. Toward chemical accuracy at low computational cost: Density-functional theory with σ-functionals for the correlation energy. *The Journal of Chemical Physics*, **2021**, 154 (1), 014104. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0026849/15583499/014104_1_online.pdf, DOI: 10.1063/5.0026849.

PAPERS:

- Langreth, C.; Perdew, J.P. Solid. State. Comm., **1975**, 17(11), 1425-1429 The Exchange-Correlation Energy of a Metallic Surface [328]
- Furche, F. Phys. Rev. B, **2001**, 64, 195120, Molecular tests oft he Random Phase Approximation tot he Exchange-Correlation Energy Functional [329]
- Furche, F. J. Chem. Phys. **2008**, 129, 114105 Develpping the Random Phase Approximation into a Practical post-Kohn-Sham Correlation Model [330]
- Trushin, E.; Thierbach, A., Görling, A. **2021**, J. Chem. Phys., 154, 014104, Toward Chemical Accuracy at Low Computational Cost: Density-Fnctional Theory with sigma-Functionals for the Correlation Energy [331]

3.9 Perturbation Theory - MP2

In this section, we discuss the second order Møller-Plesset Perturbation Theory (MP2) and its implementation in the `orca_mp2` module. Higher order correlated methods, such as MP3 and MP4 theories, are available through computer generated code in the *AUTOCI module*.

3.9.1 Standard (non-RI) MP2

The standard (or full accuracy) MP2 module has two different branches. One branch is used for energy calculations, the other for gradient calculations.

For standard MP2 energies, the program performs two half-transformations and the half-transformed integrals are stored on disk in compressed form. This appears to be the most efficient approach that can also be used for medium sized molecules.The module should parallelize acceptably well as long as I/O is not limiting.

For standard MP2 gradients, the program performs four quarter transformations that are ordered by occupied orbitals. Here, the program massively benefits from large core memory (`%maxcore`) since this minimizes the number of batches that are to be done. I/O demands are minimal in this approach.

In "memory mode" (Q1Opt>0) basically the program treats batches of occupied orbitals at the same time. Thus, there must be at least enough memory to treat a single occupied MO at each pass. Otherwise the MP2 module will fail. Thus, potentially, MP2 calculations on large molecules take significant memory and may be most efficiently done through the RI approximation.

Alternatively, in the "disk based mode" (Q1Opt=-1) the program performs a half transformation of the exchange integrals and stores the transformed integrals on disk. A bin-sort then leads to the AO operator $K^{ij}(\mu,\nu) = (i\mu|j\nu)$ in (11|22) integral notation. These integrals are then used to make the final K^{ij}(a,b) (a,b=virtual MOs) and the EMP2 pair energy contributions. In many cases, and in particular for larger molecules, this algorithm is much more efficient than the memory based algorithm. It depends, however, much more heavily on the I/O system of the computer that you use. It is important, that the program uses the flags `CFLOAT, UCFLOAT, CDOUBLE` or `UCDOUBLE` in order to store the unsorted and sorted AO exchange integrals. Which flag is used will influence the performance of the program and to some extent the accuracy of the result (float based single precision results are usually very slightly less

accurate; μEh-range deviations from the double precision result[1]). Finally, gradients are presently only available for the memory based algorithm since in this case a much larger set of integrals is required.

The `! MP2` command does the following: (a) it changes the `Method` to `HFGTO` and (b) it sets the flag `DoMP2` to `true`. The program will then first carry out a Hartree-Fock SCF calculation and then estimate the correlation energy by MP2 theory. RHF, UHF and high-spin ROHF reference wavefunctions are permissible and the type of MP2 calculation to be carried out (for high-spin ROHF the gradients are not available) is automatically chosen based on the value of `HFTyp`. If the SCF is carried out conventionally, the MP2 calculation will also be done in a conventional scheme unless the user forces the calculation to be direct. For `SCFMode =Direct` the MP2 energy evaluation will be fully in the integral direct mode.

The following variables can be adjusted in the block for conventional MP2 calculations:

```
%mp2
  EMin       -1.5       # orbital energy cutoff that defines the
                        # frozen core in Eh
  EMax        1.0e3     # orbital energy cutoff that defines the
                        # neglected virtual orbitals in Eh
  EWin        EMin,EMax # the same, but accessed as array
                        # (respects settings in %method block!)
  MaxCore     256       # maximum amount of memory (in MB) to be
                        # used for integral buffering
  ForceDirect false     # Force the calculation to be integral
                        # direct
  RI          false     # use the RI approximation
  F12         false     # apply F12 correction
  Q1Opt                 # For non-RI calculations a flag how to perform
                        # the first quarter transformation
                        #  1 - use double precision buffers
                        #      (default for gradient runs)
                        #  2 - use single precision buffers. This reduces
                        #      the memory usage in the bottleneck step by
                        #      a factor of two. If several passes are re-
                        #      quired, the number of passes is reduced by
                        #      a factor of two.
                        # -1 - Use a disk based algorithm. This respects
                        #      the flags UCFLOAT,CFLOAT,UCDOUBLE and
                        #      CDOUBLE. (but BE CAREFUL with FLOAT)
                        #      (default for energy runs)
  PrintLevel  2         # How much output to produce. PrintLevel 3 produces
                        # also pair correlation energies and other info.
  DoSCS       false     # use spin-component scaling
  Ps          1.2       # scaling factor for ab pairs
  Pt          0.333     # scaling factor for aa and bb pairs
  Density     none      # no density construction
              unrelaxed # only "unrelaxed densities"
              relaxed   # full relaxed densities
  NatOrbs     false     # calculate natural orbitals

  # Generate guess natural orbitals for CASSCF:
  # "TNat" is an alias for natural orbitals with unrelaxed density
  # omitting amplitudes < TNat. (Default = no truncation).
  TNat        0.0       # good results are obtained with 1e-6
```

Note

Throughout this section, indices $i, j, k, \ldots$ refer to occupied orbitals in the reference determinant, $a, b, c, \ldots$ to virtual orbitals and $p, q, r, \ldots$ to general orbitals from either set while $\mu, \nu, \kappa, \tau, \ldots$ refer to basis functions.

[1] However, sometimes, and in particular when transition metals and core orbitals are involved we have met unpleasantly large errors. So – be careful and double check when using floats!

3.9.2 RI-MP2

The RI-MP2 module is of a straightforward nature. The program first transforms the three-index integrals $(ia|\tilde{P})$, where "i" is a occupied, "a" is a virtual MO and "$\tilde{P}$" is an auxiliary basis function that is orthogonalized against the Coulomb metric. These integrals are stored on disk, which is not critical, even if the basis has several thousand functions. The integral transformation is parallelized and has no specifically large core memory requirements.

In the next step, the integrals are read ordered with respect to the occupied labels and the exchange operators $K^{ij}(a,b) = (ia|jb) = \sum_{\tilde{P}}^{\text{NAux}}(ia|\tilde{P})(\tilde{P}|jb)$ are formed in the rate limiting O(N^5) step. This step is done with high efficiency by a large matrix multiplication and parallelizes well. From the exchange operators, the MP2 amplitudes and the MP2 energy is formed. The program mildly benefits from large core memory (%maxcore) as this minimizes the number of batches and hence reads through the integral list.

The RI-MP2 gradient is also available. Here, all necessary intermediates are made on the fly.

In the RI approximation, one introduces an auxiliary fitting basis $\eta_P(\mathbf{r})$ and then approximates the two-electron integrals in the Coulomb metric as:

$$(pq|rs) \approx \sum_{PQ} (pq|P)\, V_{PQ}^{-1}\, (Q|rs) \tag{3.24}$$

where $V_{PQ} = (P|Q)$ is a two-index electron-electron repulsion integral. As first discussed by Weigend and Häser, the closed-shell case RI-MP2 gradient takes the form:

$$E^x_{\text{RI-MP2}} = 2\sum_{\mu\nu P} (\mu\nu|P)^{(x)} \sum_i c_{\mu i}\Gamma^P_{i\nu} + \sum_{RS} V^x_{RS}\left(\mathbf{V}^{-1/2}\gamma\mathbf{V}^{-1/2}\right)_{RS} + \langle \mathbf{DF}^x\rangle \tag{3.25}$$

The **F**-matrix derivative terms are precisely handled as in the non-RI case and need not be discussed any further. Γ^P_{ia} is a three-index two-particle "density":

$$\Gamma^P_{ia} = \sum_{jbQ} (1+\delta_{ij})\, \tilde{t}^{ij}_{ab} V^{-1/2}_{PQ}\, (Q|jb) \tag{3.26}$$

Which is partially transformed to the AO basis by:

$$\Gamma^P_{i\nu} = \sum_a c_{\nu a}\Gamma^P_{ia} \tag{3.27}$$

The two-index analogue is given by:

$$\gamma_{PQ} = \sum_{iaR} \Gamma^Q_{ia}\, (ia|R)\, V^{-1/2}_{RP} \tag{3.28}$$

The RI contribution to the Lagrangian is particularly convenient to calculate:

$$L^{RI}_{ai} = \sum_\mu c_{\mu a}\left[2\sum_{PQ\nu} \Gamma^P_{i\nu}\, (\mu\nu|Q)\, V^{-1/2}_{PQ}\right] \tag{3.29}$$

In a similar way, the remaining contributions to the energy weighted density matrix can be obtained efficiently. Note, however, that the response operator and solution of the CP-SCF equations still proceed via traditional four- index integrals since the SCF operator was built in this way. Thus, while the derivatives of the three-index integrals are readily and efficiently calculated, one still has the separable contribution to the gradient, which requires the derivatives of the four-index integrals.

The RI-MP2 energy and gradient calculations can be drastically accelerated by employing the RIJCOSX or the RIJDX approximation.

3.9.3 Calculating MP2 and RI-MP2 Energies

You can do conventional or integral direct MP2 calculations for RHF, UHF or high-spin ROHF reference wavefunctions. MP3 functionality is not implemented as part of the MP2 module, but can be accessed through the MDCI module. Analytic gradients are available for RHF and UHF. The analytic MP2-Hessians have been deprecated with ORCA-6.0. The frozen core approximation is used by default. For RI-MP2 the $\langle \hat{S}^2 \rangle$ expectation value is computed in the unrestricted case according to [332]. An extensive coverage of MP2 exists in the literature.[180, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345]

```
! MP2 def2-TZVP TightSCF
%paras  rCO = 1.20
        ACOH = 120
        rCH  = 1.08
        end
* int 0 1
C 0 0 0 0.00 0.0 0.00
O 1 0 0 {rCO} 0.0 0.00
H 1 2 0 {rCH} {ACOH} 0.00
H 1 2 3 {rCH} {ACOH} 180.00
*
```

Note

There are two algorithms for MP2 calculations without the RI approximation. The first one uses main memory as much as possible. The second one uses more disk space and is usually faster (in particular, if you run the calculations in single precision using `! FLOAT, UCFLOAT or CFLOAT`). The memory algorithm is used by specifying `Q1Opt >0` in the `%mp2` block whereas the disk based algorithm is the default or specified by `Q1Opt = -1`. Gradients are presently only available for the memory based algorithm.

The RI approximation to MP2[342, 343, 344, 345] is fairly easy to use, too. It results in a tremendous speedup of the calculation, while errors in energy differences are very small. For example, consider the same calculation as before:

```
# only the auxiliary basis set def2-TZVP/C is added to
# the keyword line
#
! RI-MP2 def2-TZVP def2-TZVP/C TightSCF
%mp2     MaxCore 100
         end
%paras   rCO  =  1.20
         ACOH =  120
         rCH  =  1.08
         end
* int 0 1
 C  0 0 0 0.00   0.0     0.00
 O  1 0 0 {rCO}  0.0     0.00
 H  1 2 0 {rCH} {ACOH}   0.00
 H  1 2 3 {rCH} {ACOH}   180.00
*
```

Generally, the RI approximation can be switched on by setting `RI true` in the `%mp2` block. Specification of an appropriate auxiliary basis set ("/C") for correlated calculations is required. Note that if the RIJCOSX method (section *RIJCOSX*) or the RI-JK method (section *RI-JK*) is used to accelerate the SCF calculation, then two basis sets should be specified: firstly the appropriate Coulomb ("/J") or exchange fitting set ("/JK"), and secondly the correlation fitting set ("/C"), as shown in the example below.

```
# Simple input line for RIJCOSX:
! RHF RI-MP2 RIJCOSX def2-TZVP def2/J def2-TZVP/C TightSCF

# Simple input line for RI-JK:
```

(continues on next page)

(continued from previous page)

```
! RHF RI-MP2 RI-JK def2-TZVP def2/JK def2-TZVP/C TightSCF
```

The MP2 module can also do Grimme's spin-component scaled MP2 [346]. It is a semi-empirical modification of MP2 which applies different scaling factors to same-spin and opposite-spin components of the MP2 energy. Typically it gives fairly bit better results than MP2 itself.

```
#
# Spin-component scaled MP2 example
#
! SCS-MP2 def2-TZVPP TightSCF
%paras   rCO  =  1.20
         ACOH =  120
         rCH  =  1.08
         end
* int 0 1
 C  0 0 0 0.00   0.0     0.00
 O  1 0 0 {rCO}  0.0     0.00
 H  1 2 0 {rCH}  {ACOH}  0.00
 H  1 2 3 {rCH}  {ACOH}  180.00
*
```

Energy differences with SCS-MP2 appear to be much better than with MP2 itself according to Grimme's detailed evaluation study. For the sake of efficiency, it is beneficial to make use of the RI approximation using the `RI-SCS-MP2` keyword. The opposite-spin and same-spin scaling factors can be modified using `PS` and `PT` in the `%mp2` block, respectively. By default, `PS` $= 6/5$ and `PT` $= 1/3$.

> **Note**
>
> In very large RI-MP2 runs you can cut down the amount of main memory used by a factor of two if you use the keyword `! FLOAT`. This is more important in gradient runs than in single point runs. Deviations from double precision values for energies and gradients should be in the μEh and sub-μEh range. However, we have met cases where this option introduced a large and unacceptable error, in particular in transition metal calculations. You are therefore advised to be careful and check things out beforehand.

A word of caution is due regarding MP2 calculations with a linearly dependent basis. This can happen, for example, with very diffuse basis sets (see *Linear Dependence* for more information). If some vectors were removed from the basis in the SCF procedure, those redundant vectors are still present as "virtual" functions with a zero orbital energy in the MP2 calculation. When the number of redundant vectors is small, this is often not critical (and when their number is large, one should probably use a different basis). However, it is better to avoid linearly dependent basis sets in MP2 calculations whenever possible. Moreover, in such a situation the orbitals should not be read with the `MORead` and `NoIter` keywords, as that is going to produce wrong results!

3.9.4 Frozen Core Options

In MP2 energy and gradient runs the Frozen Core (FC) approximation is applied by default. This implies that the core electrons are not included in the perturbation treatment, since the inclusion of dynamic correlation in the core electrons usually effects relative energies or geometry parameters insignificantly.

The frozen core option can be switched on or off with `FrozenCore` or `NoFrozenCore` in the simple input line. Furthermore, frozen orbitals can be selected by means of an energy window:

```
%method FrozenCore FC_EWIN end
%mp2 ewin -1.5, 1.0e3 end
```

More information and the different options can be found in section *Frozen Core Options*

3.9.5 MP2 and RI-MP2 Gradients

Geometry optimization with MP2, RI-MP2, SCS-MP2 and RI-SCS-MP2 proceeds just as with any SCF method. With frozen core orbitals, second derivatives of any kind are currently only available numerically. The RIJCOSX approximation (section *RIJCOSX*) is supported in RI-MP2 and hence also in double-hybrid DFT gradient runs (it is in fact the default for double-hybrid DFT since ORCA 5.0). This leads to large speedups in larger calculations, particularly if the basis sets are accurate.

```
#
# MP2 optimization example
#
! SCS-MP2 def2-TZVP OPT NoFrozenCore
* int 0 1
 C  0 0 0 0.00   0.0     0.00
 O  1 0 0 1.20   0.0     0.00
 H  1 2 0 1.09 120.0     0.00
 H  1 2 3 1.09 120.0   180.00
*
```

This job results in:

```
-----------------------------------------------------------------------------
Redundant Internal Coordinates

--- Optimized Parameters ---
(Angstroem and degrees)

Definition                    OldVal   dE/dq     Step     FinalVal
-------------------------------------------------------------------------------
1. B(O   1,C   0)                1.2081  0.000488 -0.0003    1.2078
2. B(H   2,C   0)                1.1027  0.000009 -0.0000    1.1027
3. B(H   3,C   0)                1.1027  0.000009 -0.0000    1.1027
4. A(O   1,C   0,H   3)          121.85  0.000026   -0.00    121.85
5. A(H   2,C   0,H   3)          116.29 -0.000053    0.01    116.30
6. A(O   1,C   0,H   2)          121.85  0.000026   -0.00    121.85
7. I(O   1,H   3,H   2,C   0)     -0.00 -0.000000    0.00      0.00
-------------------------------------------------------------------------------
```

Just to demonstrate the accuracy of RI-MP2, here is the result with RI-SCS-MP2 instead of SCS-MP2, with the addition of def2-TZVP/C:

```
-----------------------------------------------------------------------------
Redundant Internal Coordinates

--- Optimized Parameters ---
(Angstroem and degrees)

Definition                    OldVal   dE/dq     Step     FinalVal
-------------------------------------------------------------------------------
1. B(O   1,C   0)                1.2081  0.000487 -0.0003    1.2078
2. B(H   2,C   0)                1.1027  0.000009 -0.0000    1.1027
3. B(H   3,C   0)                1.1027  0.000009 -0.0000    1.1027
4. A(O   1,C   0,H   3)          121.85  0.000026   -0.00    121.85
5. A(H   2,C   0,H   3)          116.29 -0.000053    0.01    116.30
6. A(O   1,C   0,H   2)          121.85  0.000026   -0.00    121.85
7. I(O   1,H   3,H   2,C   0)     -0.00  0.000000   -0.00     -0.00
-------------------------------------------------------------------------------
```

You see that *nothing* is lost in the optimized geometry through the RI approximation thanks to the efficient and accurate RI-auxiliary basis sets of the Karlsruhe group (in general the deviations in the geometries between standard MP2 and RI-MP2 are very small). Thus, RI-MP2 really is a substantial improvement in efficiency over standard MP2.

Geometric gradients can be calculated with RI-MP2 in conjunction with the RIJCOSX method. They are called the same way as with a conventional SCF wave function, for example to perform a geometry optimization with tight convergence parameters: (Please note that you have to switch on NumFreq for the MP2-Hessian, as the analytical (RI-)MP2-Hessians are no longer available).

```
! RI-MP2 def2-TZVPP def2/J def2-TZVPP/C TightSCF RIJCOSX
! TightOpt
...
```

3.9.6 RIJCOSX-RI-MP2 Gradients

Additional grids are introduced for the RIJCOSX-MP2 gradient. They have sensible default settings and therefore do not usually require any intervention from the user. However, a number of expert options are available, as described below.

The COSX terms in the Z-vector equations are calculated on a grid, controlled by the keywords `Z_GridX` and `Z_IntAccX`, as discussed in sections *Changing TD-DFT, CP-SCF and Hessian Grids* and *CP-SCF Options*. For example, the `DefGrid3` CP-SCF COSX grid can be requested as:

```
%method
  Z_GridX   2     # Lebedev 110-point grid
  Z_IntAccX 3.067 # radial integration accuracy
end
```

The grid used for evaluation of the response operator on the right-hand side of the Z-vector equations (see for example eqs (3.52) and (3.53)) can be independently selected using the keyword `Z_GridX_RHS`. Note that starting with ORCA 5, the usage is different to `Z_GridX` - the choice is between one of the three grids used during the RIJCOSX SCF procedure: a small grid for the initial iterations, a medium grid for the final iterations (default in ORCA 5), and a large grid to evaluate the energy more accurately after the iterations have converged.

```
%method
  Z_GridX_RHS 1   # small SCF grid
              2   # medium SCF grid (default)
              3   # large SCF grid
end
```

Yet another grid is used to evaluate basis functions derivatives. Appropriate parameters are chosen through `! DefGridn` (in addition to the three SCF grids), but one can override this by setting the angular (`GridX`) and radial (`IntAccX`) grids explicitly through:

```
%mp2 GridX    4       # default 4: angular Lebedev grid 302
     IntAccX  4.871   # radial grid
end
```

3.9.7 MP2 and RI-MP2 Second Derivatives

Analytical second-order properties with the MP2, RI-MP2 and double-hybrid DFT methods are available in ORCA for calculations without frozen core orbitals. The most expensive term in the second derivative calculations is the four-external contribution which can be evaluated either via an AO direct (default) or a semi-numerical Chain-of-Spheres approach. In case that the latter approach is chosen, appropriate grid parameters are defined through the `! DefGridn` settings. However, a more fine-grained specification is available to expert users as follows:

```
%mp2 KCOpt _AOBLAS          # (default) AO direct with BLAS routines
           _COSX            # semi-numerical evaluation using the COSX method
     KC_GridX    2          # default 2: angular Lebedev grid 110
     KC_IntAccX  4.020      # radial grid
end
```

Alternatively, all the grid settings can be defined in the `%method` block, as discussed in section *SCF Grid Keyword List*. The first three entries define the three SCF grids, the fourth entry the MP2 grid for basis function derivatives (refer to section *RIJCOSX-RI-MP2 Gradients*) and the fifth entry the grid for the four-external contribution.

```
%method
   IntAccX     Acc1,  Acc2,  Acc3,  Acc4,   Acc5
   GridX       Ang1,  Ang2,  Ang3,  Ang4,   Ang5
end
```

3.9.8 MP2 Properties, Densities and Natural Orbitals

The MP2 method can be used to calculate electric and magnetic properties such as dipole moments, polarizabilities, hyperfine couplings, g-tensors or NMR chemical shielding tensors. For this purpose, the appropriate MP2 density needs to be requested - otherwise the properties are calculated using the SCF density!

Two types of densities can be constructed - an "unrelaxed" density (which basically corresponds to the MP2 expectation value density) and a "relaxed" density which incorporates orbital relaxation. For both sets of densities a population analysis is printed if the SCF calculation also requested this population analysis. These two densities are stored as `JobName.pmp2ur.tmp` and `JobName.pmp2re.tmp`, respectively. For the open shell case case the corresponding spin densities are also constructed and stored as `JobName.rmp2ur.tmp` and `JobName.rmp2re.tmp`.

In addition to the density options, the user has the ability to construct MP2 natural orbitals. If relaxed densities are available, the program uses the relaxed densities and otherwise the unrelaxed ones. The natural orbitals are stored as `JobName.mp2nat` which is a GBW type file that can be read as input for other jobs (for example, it is sensible to start CASSCF calculations from MP2 natural orbitals). The density construction can be controlled separately in the input file (even without running a gradient or optimization) by:

```
#
# MP2 densities and natural orbitals
#
%mp2  Density none          # no density
              unrelaxed     # unrelaxed density
              relaxed       # relaxed density
      NatOrbs true          # Natural orbital construction on or off
      end
```

Below is a calculation of the dipole and quadrupole moments of a water molecule:

```
! RI-MP2 def2-SVP def2-SVP/C
%mp2 density relaxed end
%elprop dipole     true
        quadrupole true
        end
* int 0 1
O 0 0 0 0      0      0
H 1 0 0 0.9584 0      0
H 1 2 0 0.9584 104.45 0
*
```

Another example is a simple g-tensor calculation with MP2:

```
! RI-MP2 def2-SVP def2-SVP/C TightSCF SOMF(1X) NoFrozenCore
%eprnmr gtensor  1
        ori     CenterOfElCharge
end
%mp2 density relaxed end
* int 1 2
O 0 0 0 0      0      0
H 1 0 0 1.1056 0      0
```

(continues on next page)

(continued from previous page)

```
H 1 2 0 1.1056 109.62 0
*
```

NMR chemical shielding as well as g-tensor calculations with GIAOs are only available for RI-MP2. The input for NMR chemical shielding looks as follows:

```
! RIJK RI-MP2 def2-SVP def2/JK def2-SVP/C TightSCF NMR NoFrozenCore
%mp2
  density relaxed # required
end
* int 0 1
O 0 0 0 0      0      0
H 1 0 0 1.1056 0      0
H 1 2 0 1.1056 109.62 0
*
```

Note that by default core electrons are not correlated unless the `NoFrozenCore` keyword is present.

For details, see sections *Perturbation Theory - MP2* and *MP2 Level Magnetic Properties*.

3.9.9 RI-MP2 and Double-Hybrid DFT Response Properties

Starting from ORCA 5, both the electric (for the dipole polarizability) and the magnetic (for NMR shielding and the EPR g-tensor) field response as well as the nucleus-orbit response (hyperfine couplings A_{orb} term) for RI-MP2 (and double-hybrid functionals) is handled by a different implementation of the RI-MP2 second derivatives than that used for geometric Hessian calculations (*MP2 and RI-MP2 Second Derivatives*). This code is more efficient, uses the RI approximation throughout (including the four-external contribution) and supports frozen core orbitals. The implementation is described in detail in refs [347, 348]. Consider the following input for a GIAO-RI-MP2 NMR shielding calculation:

```
! RIJK RI-MP2 def2-SVP def2/JK def2-SVP/C TightSCF NMR NoFrozenCore
%mp2
  Density         relaxed # required
  UsePertCanOrbs  true    # Whether to use perturbed canonical orbitals for
                          # the internal block of the perturbed Fock matrix
  PertCan_EThresh 1e-6    # Energy threshold for special treatment of
                          # degenerate orbital pairs
  PertCan_UThresh 10      # Coefficient threshold for special treatment of
                          # strongly interacting orbital pairs
  FCut            1e-5    # Threshold for internal perturbed Fock elements
  RespStoreT      true    # Whether to precalculate and store all necessary
                          # unperturbed amplitudes on disk
  RespDijConv     false   # Whether to store intermediates required for the
                          # internal block of the response density on disk
end
* int 0 1
O 0 0 0 0      0      0
H 1 0 0 1.1056 0      0
H 1 2 0 1.1056 109.62 0
*
```

By default perturbed canonical orbitals are used for the occupied block, i.e., the internal orbital rotation coefficients are chosen as

$$U_{ij}^{\mathbf{B}} = \frac{F_{ij}^{(\mathbf{B})} - S_{ij}^{(\mathbf{B})}\varepsilon_j}{\varepsilon_j - \varepsilon_i}$$

which results in $F_{ij}^{\mathbf{B}} = 0$, thereby eliminating its contribution to the perturbed amplitudes:

$$T_{ab}^{ij,\mathbf{B}} \leftarrow -\sum_k \left[T_{ab}^{ik} F_{kj}^{\mathbf{B}} + T_{ab}^{kj} F_{ki}^{\mathbf{B}}\right] \tag{3.30}$$

If $|\epsilon_j - \epsilon_i| <$ `PertCan_EThresh` or $|U^B_{ij}| >$ `PertCan_UThresh`, then $U^{\mathbf{B}}_{ij}$ is chosen using the standard formula

$$U^{\mathbf{B}}_{ij} = -\frac{1}{2} S^{(\mathbf{B})}_{ij}$$

And the relevant contributions to eq (3.30) are added, unless $\left|F^{\mathbf{B}}_{ij}\right| <$ `FCut`. The required amplitudes $\mathbf{T}^{ik}$ and $\mathbf{T}^{kj}$ (all amplitudes, in case `UsePertCanOrbs = false`) are stored on disk if `RespStoreT = true` or recalculated as needed otherwise. The latter option is significantly slower and not recommended unless disk space is an issue. Similarly, in the case of insufficient RAM, the option `RespDijConv = true` tells ORCA to store all amplitudes in the batch (required to calculate $D^{\mathbf{B}}_{ij}$) on disk, rather than keep them in memory. The 3-index 2-particle densities, needed for the right-hand side of the Z-vector equations, are always stored on disk.

Note also that in this implementation the RIJCOSX Fock-response terms are calculated with one of the SCF grids, chosen with `Z_GridX_RHS` (see section *CP-SCF Options*).

3.9.10 Local MP2 calculations

Purely domain-based local MP2 methodology dates back to Pulay and has been developed further by Werner, Schütz and co-workers. ORCA features a local MP2 method (DLPNO-MP2) that combines the ideas of domains and local pair natural orbitals, so that RI-MP2 energies are reproduced efficiently to within chemical accuracy. Its default thresholds are chosen to reproduce about 99.9% of the total RI-MP2 correlation energy, resulting in an accuracy of a fraction of 1 kcal/mol for energy differences. Due to the intricate connections with other DLPNO methods, reading of the section *Local correlation (DLPNO)* is recommended. A full description of the method for RHF reference wave functions has been published.[349]

The local MP2 method becomes truly beneficial for very large molecules and can be used to compute energies of systems containing several hundred atoms. Fig. 3.3 shows the scaling behavior for linear alkane chains. Note that this represents an idealized situation. For three-dimensional molecules the crossover with canonical RI-MP2 is going to occur at a later point.

Fig. 3.3: Scaling of the DLPNO-MP2 method with default thresholds for linear alkane chains in def2-TZVP basis. Shown are also the times for the corresponding Hartree-Fock calculations with RIJCOSX and for RI-MP2.

Since DLPNO-MP2 employs an auxiliary basis set to evaluate integrals, its energies converge systematically to RI-MP2 as thresholds are tightened. The computational effort of DLPNO-MP2 with default settings is usually compa-

rable with or less than that of a Hartree-Fock calculation. However, for small and medium-sized molecules, RI-MP2 is even faster than DLPNO-MP2.

Calculations on open-shell systems are supported through a UHF treatment. While most approximations are consistent between the RHF and UHF versions, this is not true for the PNO spaces. **DLPNO-MP2 gives different energies for closed-shell molecules in the RHF and UHF formalisms. When calculating reaction energies or other energy differences involving open-shell species, energies of closed-shell species must also be calculated with UHF-DLPNO-MP2, and not with RHF-DLPNO-MP2.** As for canonical MP2, ROHF reference wave functions are subject to an ROMP2 treatment through the UHF machinery. It is not consistent with the RHF version of DLPNO-MP2, unlike in the case of RHF-/ROHF-DLPNO-CCSD.

In the following, the most important design principles of the RHF-DLPNO-MP2 are pointed out:

Both DLPNO-CCSD(T) and DLPNO-MP2 are linear-scaling methods (albeit the former has a larger prefactor). This means that if a DLPNO-MP2 calculation can be performed, DLPNO-CCSD(T) is often going to be within reach, too. However, CCSD(T) is generally much more accurate than MP2 and thus should be given preference.

A correlation fitting set must be provided, as the method makes use of the RI approximation.

Canonical RI-MP2 energy differences are typically reproduced to within a fraction of 1 kcal/mol. The default thresholds have been chosen so as to reproduce about $99.9\,\%$ of the total RI-MP2 correlation energy.

The preferred way to control the accuracy of the method is by means of specifying "LoosePNO", "NormalPNO" and "TightPNO" keywords. "NormalPNO" corresponds to default settings and does not need to be given explicitly. More details and an exhaustive list of input parameters are provided in section *Local MP2 calculations*. Note that the thresholds differ from DLPNO coupled cluster.

Results obtained from RI-MP2 and DLPNO-MP2, or from DLPNO-MP2 with different accuracy settings, must never be mixed, such as when computing energy differences. In calculations involving open-shell species, even the closed-shell molecules need to be subject to a UHF treatment.

Spin-component scaled DLPNO-MP2 calculations are invoked by using the `! DLPNO-SCS-MP2` keyword instead of `! DLPNO-MP2` in the simple input line. Weights for same-spin and opposite-spin contributions can be adjusted as described for the canonical SCS-MP2 method. Likewise, there is a `DLPNO-SOS-MP2` keyword to set the parameters defined by the SOS-MP2 method (but there is no Laplace transformation involved).

The frozen core approximation is used by default. If core orbitals are involved in the calculation, they are subject to the treatment described in section *Local MP2 calculations*.

Calculations can be performed in parallel.

It may be beneficial to accelerate the Hartree-Fock calculation by means of the RIJCOSX method (requiring specification of a second auxiliary set).

Unlike in the 2013 version of the DLPNO methodology, domains are selected by means of the differential overlap $\sqrt{\int i^2(\mathbf{r})\tilde{\mu}'^2(\mathbf{r})d\mathbf{r}}$ between localized MOs i and projected atomic orbitals (PAOs) $\tilde{\mu}'$ which are normalized to unity. The default value for the corresponding cutoff is $T_{\text{CutDO}} = 10^{-2}$.

MP2 amplitudes for each pair of localized orbitals ij are expressed in a basis of pair natural orbitals (PNOs) $\tilde{a}_{ij}$. PNOs are obtained from diagonalization of an approximate, "semicanonical" MP2 pair density $\mathbf{D}^{ij}$. Only PNOs with an occupation number $> T_{\text{CutPNO}}$ are retained, with a default value of $T_{\text{CutPNO}} = 10^{-8}$ for DLPNO-MP2. The pair density is given by:

$$\mathbf{D}^{ij} = \mathbf{T}^{ij\dagger}\tilde{\mathbf{T}}^{ij} + \mathbf{T}^{ij}\tilde{\mathbf{T}}^{ij\dagger} \quad \text{where} \quad \begin{aligned} T^{ij}_{\tilde{\mu}\tilde{\nu}} &= -\frac{(i\tilde{\mu}|j\tilde{\nu})}{\varepsilon_{\tilde{\mu}} + \varepsilon_{\tilde{\nu}} - F_{ii} - F_{jj}} \\ \tilde{\mathbf{T}}^{ij} &= (1+\delta_{ij})^{-1}\left(4\mathbf{T}^{ij} - 2\mathbf{T}^{ij\dagger}\right) \end{aligned}$$

Since the occupied block of the Fock matrix is not diagonal in the basis of localized orbitals, the MP2 amplitudes $\mathbf{T}^{ij}$ are obtained by solving the following set of residual equations iteratively (where subscripts of PNOs have been dropped):

$$R^{ij}_{\tilde{a}\tilde{b}} = \left(i\tilde{a}\middle|j\tilde{b}\right) + \left(\varepsilon_{\tilde{a}} + \varepsilon_{\tilde{b}} - F_{ii} - F_{jj}\right)T^{ij}_{\tilde{a}\tilde{b}} - \sum_{k\neq i}\sum_{\tilde{c}\tilde{d}} F_{ik} S^{ij,kj}_{\tilde{a}\tilde{c}} T^{kj}_{\tilde{c}\tilde{d}} S^{kj,ij}_{\tilde{d}\tilde{b}} - \sum_{k\neq j}\sum_{\tilde{c}\tilde{d}} F_{kj} S^{ij,ik}_{\tilde{a}\tilde{c}} T^{ik}_{\tilde{c}\tilde{d}} S^{ik,ij}_{\tilde{d}\tilde{b}} = 0$$

The largest part of the error relative to canonical RI-MP2 is controlled by the domain (`TCutDO`) and PNO (`TCutPNO`) thresholds, which should be adequate for most applications. If increased accuracy is needed (e.g. for

studying weak interactions), tighter truncation criteria can be invoked by means of the `! TightPNO` keyword. Conversely, a less accurate but faster calculation can be performed with the `! LoosePNO` keyword. For more details refer to table Table 3.33.

Fitting domains are determined by means of orbital Mulliken populations with a threshold $T_{\text{CutMKN}} = 10^{-3}$. This threshold results in an error contribution that is typically about an order of magnitude smaller than the overall total energy deviation from RI-MP2.

Prior to performing the local MP2 calculation, pairs of localized molecular orbitals ij are prescreened using an MP2 energy estimate with a dipole approximation, and the differential overlap integral between orbitals i and j. This procedure has been chosen conservatively and leads to minimal errors.

Residual evaluation can be accelerated significantly by neglecting terms with associated Fock matrix elements F_{ik} and F_{kj} below $F_{\text{Cut}} = 10^{-5}$. Errors resulting from this approximation are typically below 1 μE_h and thus negligible.

Sparsity of the MO and PAO coefficient matrices in atomic orbital basis is exploited to accelerate integral transformations for large systems. Truncation of the coefficients is controlled by a parameter T_{CutC}. Neglect of these coefficients has to be performed very carefully in order to avoid uncontrollable errors. The threshold has been chosen so as to make the errors essentially vanish.

By default, core orbitals are frozen in the MP2 module. However, if core orbitals are subject to an MP2 treatment, it is necessary to use a tighter PNO cutoff for pairs involving at least one core orbital. For this purpose core orbitals and valence orbitals are localized separately. The cutoff for pairs involving core orbitals is given by $T_{\text{CutPNO}} \times T_{\text{ScalePNO_Core}}$, where $T_{\text{ScalePNO_Core}} = 10^{-2}$ by default. For more details refer to section *Including (semi)core orbitals in the correlation treatment*.

The UHF-DLPNO-MP2 implementation differs somewhat from the RHF case, particularly regarding construction of PNOs, as described below.

A separate set of PAOs $\tilde{\mu}'_\alpha$ and $\tilde{\mu}'_\beta$ is obtained for each spin case.

For $\alpha\beta$ pairs, separate pair domains of PAOs need to be determined for each spin case. For example, the α pair domain $[i_\alpha j_\beta]_\alpha$ is the union of the domains $[i_\alpha]_\alpha$ and $[j_\beta]_\alpha$. The latter domain $[j_\beta]_\alpha$ is determined by evaluating the spatial differential overlap between j_β and α-spin PAOs $\tilde{\mu}'_\alpha$.

One set of PNOs is needed for each same-spin pair. Opposite-spin pairs require a set of α-PNOs and a set of β-PNOs. In total this results in four types of PNO sets.

Semicanonical amplitudes are obtained as follows, where i, j are spin orbitals and $\tilde{\mu}\tilde{\nu}$ are nonredundant spin PAOs.

$$T^{ij}_{\tilde{\mu}\tilde{\nu}} = -\frac{\langle ij||\tilde{\mu}\tilde{\nu}\rangle}{\varepsilon_{\tilde{\mu}} + \varepsilon_{\tilde{\nu}} - F_{ii} - F_{jj}}$$

In the same-spin case $\langle i_\alpha j_\alpha||\tilde{\mu}_\alpha\tilde{\nu}_\alpha\rangle = \langle ij|\tilde{\mu}\tilde{\nu}\rangle - \langle ij|\tilde{\nu}\tilde{\mu}\rangle$, while in the opposite-spin case $\langle i_\alpha j_\beta||\tilde{\mu}_\alpha\tilde{\nu}_\beta\rangle = \langle ij|\tilde{\mu}\tilde{\nu}\rangle$.

For opposite-spin pairs, α-PNOs and β-PNOs are obtained from diagonalisation of $\mathbf{T}^{ij}\mathbf{T}^{ij\dagger}$ and $\mathbf{T}^{ij\dagger}\mathbf{T}^{ij}$, respectively. For same-spin pairs the pair density is symmetric and only one set of PNOs is needed. PNOs are discarded whenever the absolute value of their natural occupation number is below the threshold T_{CutPNO}.

The following residual equations need to be solved for the cases $R^{i_\alpha j_\alpha}_{\tilde{a}_\alpha\tilde{b}_\alpha}$, $R^{i_\beta j_\beta}_{\tilde{a}_\beta\tilde{b}_\beta}$ and $R^{i_\alpha j_\beta}_{\tilde{a}_\alpha\tilde{b}_\beta}$:

$$\begin{aligned} R^{i_\sigma j_\tau}_{\tilde{a}_\sigma\tilde{b}_\tau} &= \left\langle ij \middle|\middle| \tilde{a}\tilde{b} \right\rangle + \left(\varepsilon_{\tilde{a}} + \varepsilon_{\tilde{b}} - F_{ii} - F_{jj}\right) T^{ij}_{\tilde{a}\tilde{b}} \\ &\quad - \sum_{k_\sigma \neq i_\sigma} \sum_{\tilde{c}_\sigma \tilde{d}_\tau} F_{ik} S^{ij,kj}_{\tilde{a}\tilde{c}} T^{kj}_{\tilde{c}\tilde{d}} S^{kj,ij}_{\tilde{d}\tilde{b}} - \sum_{k_\tau \neq j_\tau} \sum_{\tilde{c}_\sigma \tilde{d}_\tau} F_{kj} S^{ij,ik}_{\tilde{a}\tilde{c}} T^{ik}_{\tilde{c}\tilde{d}} S^{ik,ij}_{\tilde{d}\tilde{b}} = 0 \end{aligned}$$

Most approximations are consistent between the RHF and UHF schemes, with the exception of the PNO truncation. This means that results would match for closed-shell molecules with $T_{\text{CutPNO}} = 0$ (provided both Hartree-Fock solutions are identical), but this is not true whenever the PNO space is truncated. Therefore, UHF-DLPNO-MP2 energies should only be compared to other UHF-DLPNO-MP2 energies, even for closed-shell species.

We found that it is necessary to use tighter PNO thresholds for UHF-DLPNO-MP2. With NormalPNO settings the default value is $T_{\text{CutPNO}} = 10^{-9}$. For an overview of accuracy settings refer to table Table 3.33. As in the RHF implementation, the PNO cutoff for pairs involving core orbitals is scaled with $T_{\text{ScalePNO_Core}}$.

Input for DLPNO-MP2 requires little specification from the user:

```
# DLPNO-MP2 calculation with standard settings
# sufficient for most purposes
! def2-TZVP def2-TZVP/C DLPNO-MP2 TightSCF

# OR: DLPNO-MP2 with tighter thresholds
# May be interesting for weak interactions, calculations with diffuse basis sets␣
↪etc.
! def2-TZVP def2-TZVP/C DLPNO-MP2 TightPNO TightSCF

%maxcore 2000

*xyz 0 1
... (coordinates)
*
```

Explicit correlation has been implemented in the DLPNO-MP2-F12 methodology for RHF reference wave functions.[350] The available approaches are C (keyword `! DLPNO-MP2-F12`) and the somewhat more approximate D (keyword `! DLPNO-MP2-F12/D`). Approach D is generally recommended as it results in a significant speedup while leading only to small errors relative to approach C. In addition to the MO and correlation fitting sets, a CABS basis set is also required for both F12 approaches as shown below.

```
# DLPNO-MP2-F12 calculation using approach C
! cc-pVDZ-F12 aug-cc-pVDZ/C cc-pVDZ-F12-CABS DLPNO-MP2-F12 TightSCF

# OR: DLPNO-MP2-F12 calculation using approach D (recommended)
! cc-pVDZ-F12 aug-cc-pVDZ/C cc-pVDZ-F12-CABS DLPNO-MP2-F12/D TightSCF
```

Table 3.33: Accuracy settings for DLPNO-MP2.

Setting	T_{CutDO}	T_{CutPNO} (RHF)	T_{CutPNO} (UHF)
LoosePNO	2×10^{-2}	10^{-7}	10^{-8}
NormalPNO	1×10^{-2}	10^{-8}	10^{-9}
TightPNO	5×10^{-3}	10^{-9}	10^{-10}

Options specific to DLPNO-MP2 are listed below.

```
%mp2 DLPNO            false  # Do DLPNO-MP2 (also requires RI true)
     TolE             1e-6   # Energy convergence threshold. Default: TolE of SCF
     TolR             5e-6   # Residual convergence threshold. Default: 5 * TolE
     MaxPNOIter       100    # Maximum number of residual iterations
     MaxLocIter       128    # Maximum number of iterations for orbital␣
↪localization
     LocMet           AHFB   # Localization method
                             # options: FB, PM, IAOIBO, IAOBOYS, NEWBOYS, AHFB
     LocTol           1e-6   # Localization convergence tolerance
                             # Default: 0.1 * TolG from SCF
     DIISStart_PNO    0      # First iteration to invoke DIIS extrapolation
     MaxDIIS_PNO      7      # length of DIIS vector
     Damp1_PNO        0.5    # Damping before DIIS is started
     Damp2_PNO        0.0    # Damping with DIIS
     MP2Shift_PNO     0.2    # level shift in amplitude update (Eh)
# Truncation parameters:
     TCutPNO          1e-8   # PNO occupation number cutoff (RHF)
                      1e-9   # PNO occupation number cutoff (UHF)
     TScalePNO_Core   1e-2   # Core PNO scaling factor
     TCutDO           1e-2   # Differential overlap cutoff for domain selection
     TCutMKN          1e-3   # Mulliken population cutoff for fitting domain␣
↪selection
     FCut             1e-5   # Occupied Fock matrix element cutoff
     TCutPre          1e-6   # Energy threshold for dipole prescreening
```

(continues on next page)

(continued from previous page)

```
    TCutDOij         1e-5   # Maximum differential overlap between screened MOs
    TCutDOPre        3e-2   # Cutoff to select PAOs for domains in prescreening
    TCutC            1e-3   # Cutoff for PAO coefficient truncation
    ScaleTCutC_MO    1.0    # Cutoff for MO truncation: TCutC * ScaleTCutC_MO
    PAOOverlapThresh 1e-8   # Threshold for constructing non-redundant PAOs
end
```

3.9.11 Local MP2 gradients

The analytical gradient has been implemented for the RHF variant of the DLPNO-MP2 method.[351, 352] It is a complete derivative of all components in the DLPNO-MP2 energy, and the results are therefore expected to coincide with numerical derivatives of DLPNO-MP2 (minus the noise).

General remarks:

No gradient is presently implemented for the UHF-DLPNO-MP2 variant.

Spin-component scaled MP2 is supported by the gradient.

Double-hybrid density functionals are supported by the gradient.

Only Foster-Boys localization is presently supported. The default converger is `AHFB` with a convergence tolerance that is automatically bound by a constant factor to the SCF orbital gradient tolerance. Using a different converger is possible, but discouraged, as the orbital localization needs to be sufficiently tightly converged.

When calculating properties without the full nuclear gradient, the relaxed MP2 density should be requested.

A number of points regarding geometry optimizations (not all of them specific to DLPNO-MP2) are worth keeping in mind:

As of 2018, we expect that the DLPNO-MP2 gradient can most beneficially be used for geometry optimizations of systems containing around 70-150 atoms. It may be faster than RI-MP2 even for systems containing 50-60 atoms or less, but the timing difference is probably not going to be very large. Of course, structures containing 200 atoms and above can (and have been) optimized, but this may take long if many geometry steps are required. On the other hand, single point gradient or density calculations can be performed for systems containing many hundred atoms.

DLPNO-MP2 is a substantially more expensive method for geometry optimizations than GGA or hybrid DFT functionals. Therefore, it is generally a good idea to start a geometry optimization with a structure that is already optimized at dispersion-corrected DFT level.

RIJCOSX can be used to accelerate exchange evaluation substantially. However, great care needs to be exercised with the grid settings. Insufficiently large grids may lead, for example, to non-planar distortions of planar molecules. The updated default grids in ORCA 5 (`DefGrid2-3`) should be sufficiently accurate to optimize neutral main group compounds. We therefore recommend these grids for general use with some careful checking in more complicated cases. Even with these grids, the calculation is a lot faster than "regular" Hartree-Fock with basis sets of triple zeta quality (or larger).

Using RIJONX is also possible.

Sufficiently large grids should be used for the exchange-correlation functional of double hybrids. The SCF calculation takes only a fraction of the time that is needed for DLPNO-MP2, and sacrificing quality because of an insufficiently accurate grid is a waste of computer time.

Optimization of large structures is often a challenge for the geometry optimizer. It may help to change the trust radius settings, to modify the settings of the `AddExtraBonds` feature, or to change other settings of the geometry optimizer. Sometimes it may be beneficial to check the geometry optimizer settings with a less demanding electronic structure method.

Finally, problems with a geometry optimization may in some cases indeed be caused by the DLPNO approximations. Using `LoosePNO` for accurate calculations is not recommended anyway, and difficulties with `NormalPNO` settings are possibly rectified by switching to `TightPNO`.

During the development process, a number of difficulties were encountered related to the orbital localization Z-vector equations. Great care was taken to work around these problems and to make the procedures as robust as possible, but a number of settings can be changed. For more information on these aspects, we recommend consulting the full paper on the DLPNO-MP2 gradient [352].

Several different solvers are implemented for the orbital localization Z-vector equations. The default is an iterative conjugate gradient solver. As an alternative, the DIIS-accelerated Jacobi solver can be used, but it tends to be inferior to the conjugate gradient solver. Moreover, a direct solver is available as a fail-safe alternative for smaller systems. As the dimension of the linear equation system is $n(n-1)/2$ for n occupied orbitals, the memory requirement and FLOP count increase as $O(n^4)$ and $O(n^6)$, respectively, and using the direct solver becomes unfeasible for large systems.

Settings for the CPSCF solver are specified the same way as for canonical MP2.

Under specific circumstances, the orbital Hessian of the orbital localization function may have zero or near-zero eigenvalues, which can lead to singular localization Z-vector equations. In particular, it is typically a consequence of continuously degenerate localized orbitals, which may (but do not need to) appear in some molecular symmetries.[353] A typical symptom are natural occupation number above 2 and below 0 for systems that would be expected to have MP2 density eigenvalues between 2 and 0 without the DLPNO approximations.

In order to work around the aforementioned problem, a procedure has been implemented to eliminate singular or near-singular eigenvectors of the localization function orbital Hessian. Vectors with an eigenvalue smaller than `ZLoc_EThresh` (or `ZLoc_EThresh_core` for the core orbitals) are subject to the modified procedure. If the program eliminates any eigenvectors, it might sometimes be a good idea to check if calculated properties are reasonable (or at least to check the natural occupation numbers). Eigenvectors of the Hessian are calculated by Davidson diagonalization by default, but direct diagonalization can be chosen for smaller systems, instead.

Diagonalization of the localization orbital Hessian can be switched off altogether by setting `ZLoc_EThresh` to 0.

If the "Asymmetric localization equation residual norm" exceeds the localization Z-vector equation tolerance (`ZLoc_Tol`), there are typically two plausible reasons: (1) the localized orbitals are not sufficiently tightly converged (too large `LocTol`) or unconverged, or (2) the orbital localization Hessian has got small eigenvalues that were not eliminated.

Usage is as simple as that of RI-MP2. For example, the following input calculates the gradient and the natural orbitals:

```
! DLPNO-MP2 def2-SVP def2-SVP/C TightSCF EnGrad
%MaxCore 512
# With 'EnGrad', specifying 'density relaxed' is unnecessary.
# However, it is needed when calculating properties without the gradient.
%MP2 Density Relaxed
     NatOrbs True
     End
*xyz 0 1
C   0.000   0.000   0.000
O   0.000   0.000   1.162
O   0.000   0.000  -1.162
*
```

The implementation supports spin-component scaling and can be used together with double-hybrid density functionals. The latter are invoked with the name of the functional preceded by "`DLPNO-`". A simple geometry optimization with a double-hybrid density functional is illustrated in the example below:

```
! DLPNO-B2PLYP D3 NormalPNO def2-TZVP def2-TZVP/C Opt
%MaxCore 1000
*xyz 0 1
O   0.000   0.000   0.000
H   0.000   0.000   1.000
H   0.000   1.000   0.000
*
```

For smaller systems, the performance difference between DLPNO-MP2 and RI-MP2 is not particularly large, but very substantial savings in computational time over RI-MP2 can be achieved for systems containing more than approximately 70-80 atoms.

Since MP2 is an expensive method for geometry optimizations, it is generally a good idea to use well-optimized starting structures (calculated, for example, with a dispersion-corrected DFT functional). Moreover, it is highly advisable to employ accurate Grids for RIJCOSX or the exchange-correlation functional (if applicable), as the SCF iterations account only for a fraction of the overall computational cost. If calculating calculating properties without requesting the gradient, `Density Relaxed` needs to be specified in the `%MP2-block`.

Only the Foster-Boys localization scheme is presently supported by the derivatives implementation. The default localizer in DLPNO-MP2 is `AHFB`, and changing this setting is strongly discouraged, since tightly converged localized orbitals are necessary to calculate the gradient.

This is an overview over the options related to the gradient:

```
# Settings specific to the localization equation z-solver
%mp2 ZLoc_Solver        CG      # Use conjugate gradient solver (default)
                        DIR     # Use direct solver
                        JAC     # Use DIIS-accelerated Jacobi solver
     ZLoc_Tol           1.0e-3  # Residual convergence tolerance for the
                                # localization Z-solver
                                # Default: same value as Z_Tol for CPSCF
     ZLoc_MaxIter       1024    # Maximum localization Z-solver iterations
     ZLoc_MaxDIIS       10      # Number of DIIS vectors for the Jacobi solver
     ZLoc_Shift         0.2     # Shift for the Jacobi solver
# Options for eliminating (near-)singular eigenvectors of the
# orbital Hessian of the localization function.
     ZLoc_EThresh       3.0e-4  # Eigenvectors with an eigenvalue below
                                # this threshold are eliminated.
     ZLoc_EThresh_core  3.0e-4  # Same as ZLoc_EThresh, but for the core orbitals.
                                # Default: identical value as ZLoc_EThresh.
# Options for determining eigenvectors of the localization orbital Hessian.
     ZLoc_UseDavidson   True    # Use Davidson diagonalization.
                                # If false, use direct diagonalization.
     ZLoc_DVDRoots      32      # Number of Davidson roots to be determined.
     ZLoc_DVDNIter      256     # Number of Davidson iterations.
     ZLoc_DVDTolE       3.0e-10 # Eigenvalue tolerance for the Davidson solver.
                                # Default: 1e-6 * ZLoc_EThresh
     ZLoc_DVDTolR       1.0e-7  # Residual tolerance for the Davidson solver.
                                # Default: 0.1 * (ZLoc_Tol)^2
     ZLoc_DVDMaxDim     10      # During Davidson diagonalization, the space of␣
↪trial
                                # vectors is expanded up to MaxDim * DVDRoots.
# Choice of the PNO processing algorithm.
     DLPNOGrad_Opt      AUTO      # Chooses automatically between RAM and DISK
                                  # (default and recommended)
                        RAM       # Enforce memory-based one-pass algorithm
                        DISK      # Enforce disk-based two-pass algorithm
                        BUFFERED  # Buffered algorithm. Usage is discouraged.
                                  # Experimental, unpredictable I/O performance.
end
```

Local MP2 Second Derivatives and Response Properties

Analytical second derivatives with respect to electric and magnetic fields are implemented for closed-shell DLPNO-MP2 (as well as double-hybrid DFT).[354] Thus, analytic dipole polarizability and NMR shielding tensors (with our without GIAOs) are available. The implementation supports spin-component scaling and double-hybrid functionals. Errors in the calculated properties are well below 0.5% when `NormalPNO` thresholds are used. Refer to section *Local MP2 Second Derivatives and Response Properties* for more information about the DLPNO-MP2 second derivatives implementation, as well as to the sections on electric (*Electrical Properties - Electric Moments and Polarizabilities*) and magnetic (*EPRNMR - keywords for magnetic properties*) properties and CP-SCF settings (*CP-SCF Options*). All considerations and options discussed in sections *Local MP2 calculations* and *Local MP2 gradients* apply here as well, while additional remarks specific to second derivatives are given below.

DLPNO-MP2 response property calculations are expected to be faster than the RI-MP2 equivalents for systems larger than about 70 atoms or 300 correlated electrons.

Using the `NormalPNO` default thresholds, relative errors in the calculated properties, due to the local approximations, are smaller than 0.5%, or 5–10 times smaller than the inherent inaccuracy of MP2 vs a more accurate method like CCSD(T).

DLPNO-MP2 second derivatives are much more sensitive to near-linear dependencies and other numerical issues than the energy or first-order properties. We have made efforts to choose reasonable and robust defaults, however we encourage the user to be critical of the results and to proceed with caution, especially if diffuse basis sets or numerical integration (DFT, COSX) are used. In the latter case, `DefGrid3` is recommended.

In particular, the near-redundancy of PAO domains introduces numerical instabilities in the algorithm. Hence, these should be truncated at `PAOOverlapThresh=1e-5`, which is higher than the default for the energy and gradient. Therefore, the energy and gradient in jobs, which include response property calculations, may deviate from jobs, which do not. The difference is much smaller than the accuracy of the method (vs RI-MP2) but it is still advisable to use the same value of `PAOOverlapThresh` in all calculations, when calculating, e.g., relative energies.

For the same reason, if diffuse basis sets are used, it is advised to set `SThresh=1e-6` in the `%scf` block.

Another instability arises due to small differences between the occupation number of kept and discarded PNOs and may result in very large errors. The smallest difference is printed during the DLPNO-MP2 relaxed density calculation:

Smallest occupation number difference between PNOs and complementary PNOs. Absolute: 3.10e-10 Relative: 3.28e-02

We found that a relative difference under 10^{-3}, which is not uncommon, may be cause for concern. To regularize the unstable equations, a level shift is applied, which is equal to T_{CutPNO} multiplied by $T_{\text{ScalePNO_LShift}}$. A reasonable value of `TScalePNO_LShift=0.1` is set by default for response property calculations but not for gradient (or energy) calculations, as these were not found to suffer from this issue, so the same considerations as above apply.

The option `DLPNOGrad_Opt=BUFFERED` is not implemented for response properties.

Below is an example for a simple DLPNO-MP2 NMR shielding calculation:

```
! DLPNO-MP2 def2-TZVP def2-TZVP/C TightSCF NMR
# MP2 relaxed density is requested automatically
*xyz 0 1
  H 0 0 0
  F 0 0 0.9
*
```

A summary of the additional options used for DLPNO-MP2 response properties is given below:

```
%mp2
     PAOOverlapThresh 1e-5   # Threshold for constructing non-redundant PAOs
                             # Default is 1e-8 for energy/gradient calculations!
     TScalePNO_LShift 0.1    # Level shift for PNO constraint equations:
                             # TScalePNO_LShift * TCutPNO
                             # Default is 0 for energy/gradient calculations!
end
```

An example input for a DSD-PBEP86 calculation of the NMR shielding and dipole polarizability tensors employing DLPNO-MP2 is given below. Note that the def2-TZVP basis set is not necessarily ideal for either shielding or polarizability.

```
! DLPNO-DSD-PBEP86/2013 D3BJ def2-TZVP def2-TZVP/C TightSCF NoFrozenCore
! RIJCOSX def2/J DefGrid3      # Use RIJCOSX with tighter grid settings
! NMR                          # Request NMR shielding
%elprop Polar 1 end            # Request polarizability
%mp2                           # These settings are default for response properties
   Density Relaxed
   PAOOverlapThresh 1e-5
   TScalePNO_LShift 0.1
```

(continues on next page)

(continued from previous page)

```
end
%eprnmr
  GIAO_2el GIAO_2el_RIJCOSX    # Also use RIJCOSX for GIAO integrals
end                            # (This is the default for !RIJCOSX)
*xyzfile 0 1 geometry.xyz
```

Numerical DLPNO-MP2 derivatives

The various truncations in local correlation methods introduce small discontinuities in the potential energy surface. For example, a small displacement may change the sizes of correlation domains, leading to a slightly larger or smaller error from the domain approximation. The default DLPNO-MP2 truncation thresholds are conservative enough, so that these discontinuities should not cause problems in geometry optimizations using analytic gradients.[352] However, if one wishes to calculate (semi-)numerical derivatives of the DLPNO-MP2 energy, gradient, or properties using finite differences, large errors can occur. Therefore, in these cases it is advisable to keep the pair lists and correlation domains fixed upon displacement. Currently, this can be achieved using the following procedure: first, the calculation at the reference geometry is carried out with the additional setting `StoreDLPNOData=true`:

```
! DLPNO-MP2 def2-SVP def2-SVP/C VeryTightSCF
%base "calc0"
%mp2
  StoreDLPNOData true
end
*xyzfile 0 1 geom0.xyz
```

This will produce the additional files *calc0.MapDLPNO00.tmp*, *calc0.MapDLPNOPre0.tmp*, *calc0.IJLIST.0.tmp*, *calc0.IJLISTSCR.0.tmp*, and *calc0.IJNPNO.0.tmp*, which are needed in the working directory for the next step together with the localized orbitals in *calc0.loc*. The calculation at the displaced geometry is then requested as:

```
! DLPNO-MP2 def2-SVP def2-SVP/C VeryTightSCF
%mp2
  RefBaseName "calc0"
end
*xyzfile 0 1 geom1.xyz
```

The program will use the orbitals from *calc0.loc* as a starting guess for the localization and map the reference orbitals to the new ones based on maximal overlap. The lists of correlated and screened out pairs are read from the files *calc0.IJLIST.0.tmp* and *calc0.IJLISTSCR.0.tmp*, while the domain information (MO-PAO, MO-Aux, etc.) is read from *calc0.MapDLPNO00.tmp* and *calc0.MapDLPNOPre0.tmp*. The number of PNOs for a each pair (stored in *calc0.IJNPNO.0.tmp*) is also kept consistent with the reference calculation: the ones with the higest occupation numbers are kept, disregarding T_{CutPNO}.

This procedure should improve the accuracy and numerical stability for manually calculated geometric derivatives of various DLPNO-MP2 properties (including those that require analytic first or second derivatives at the displaced geometries). For semi-numerical Hessian calculations (`NumFreq`), it is sufficient to add `StoreDLPNOData=true` as shown below and ORCA will handle the rest. For the sake of numerical stability, it is also recommended to increase `PAOOverlapThresh` and add a PNO level shift for the reasons described in section *Local MP2 Second Derivatives and Response Properties*.

```
! DLPNO-MP2 def2-SVP def2-SVP/C VeryTightSCF NumFreq
%mp2
  StoreDLPNOData true
  PAOOverlapThresh 1e-5
  TScalePNO_LShift 0.1
end
# geometry definition
```

Note that in case the orbital localization Hessian is (near-)singular, the mapping of orbitals from reference to displaced geometries will likely fail. No solution is presently implemented for this problem.

Multi-Level DLPNO-MP2 calculations

With the DLPNO-MP2 method it is possible to treat the interactions among different fragments of a system with varying accuracy, or exclude some interactions from the electron correlation treatment entirely. A more detailed discussion in the DLPNO-CCSD(T) context is given in section *Multi-Level Calculations* and in ref. [355]. Here we just present the technical capabilities of the MP2 module and the required input. Currently, multilayer calculations are only available for closed-shell DLPNO-MP2. Multilayer gradients and response properties are also possible. Fragments must be defined – see *Fragment Specification*.

```
! DLPNO-MP2 NoFrozenCore TightPNO
%mp2
  LoosePNOFragInter  {1 *} # * can be used as a wildcard for either or both indices
  NormalPNOFragInter {1 1} {1 2} # multiple fragment pairs can be listed like this
  TightPNOFragInter  {2 3}
  HFFragInter        {3 1} {4 2}
  CustomFragInter
    FragPairs        {4 4} {3 4} # pair definition is required
    HFOnly           false # flag to skip MP2 for these pairs - same as HFFragInter
    FrozenCore       false # flag to skip core correlation - requires !NoFrozenCore
    TCutPNO          1e-8  # custom value for these pairs
    TCutDO           1e-2  # custom value for these pairs
    TCutDOij         1e-5  # custom value for these pairs
    TCutPre          1e-6  # custom value for these pairs
  end
end
# geometry and fragment definition
```

Note that a given pair or fragments can only belong to a single layer and definitions later in the input overwrite previous ones. This means that if the above input is used in a 4-fragment calculation, the 1-4 interfragment interactions will be treated with `LoosePNO` thresholds, the interactions within fragment 1 and with fragment 2 – with `NormalPNO` thresholds, 2-3 pairs – with `TightPNO`, 1-3 and 2-4 pairs will be left at the HF level, 3-4 and 4-4 pairs will be treated with $T_{\text{CutPNO}} = 10^{-8}$ and $T_{\text{CutDO}} = 10^{-2}$ (i.e. the `NormalPNO` defaults), and 2-2 and 3-3 pairs will be left at the global (`TightPNO`) settings.

3.9.12 Explicitly correlated MP2 calculations

ORCA features an efficient explicit correlation module that is available for MP2 and coupled-cluster calculations (section *Explicitly Correlated Methods: F12-MP2 and F12-CCSD(T)*). It is described below in the context of coupled-cluster calculations.

3.9.13 MP2 in "Double-Hybrid" Density Functional Theory

A slightly more general form is met in the double-hybrid DFT gradient. The theory is briefly described below.

The energy expression for perturbatively and gradient corrected hybrid functionals as proposed by Grimme is:

$$E = V_{NN} + \left\langle \mathbf{Ph}^{+} \right\rangle + \frac{1}{2}\int\int \frac{\rho\left(\mathbf{r}_1\right)\rho\left(\mathbf{r}_2\right)}{\left|\mathbf{r}_1 - \mathbf{r}_2\right|} d\mathbf{r}_1 d\mathbf{r}_2 - \frac{1}{2} a_x \sum_{\mu\nu\kappa\tau\sigma} P^{\sigma}_{\mu\kappa} P^{\sigma}_{\nu\tau} \left(\mu\nu|\kappa\tau\right) + c_{\text{DF}} E_{\text{XC}}\left[\rho_\alpha, \rho_\beta\right] + c_{\text{PT}} E_{\text{PT}}$$

$$E = E_{\text{SCF}} + c_{\text{PT}} E_{\text{PT}} \tag{3.31}$$

Here V_{NN} is the nuclear repulsion energy and $h_{\mu\nu}$ is a matrix element of the usual one-electron operator which contains the kinetic energy and electron-nuclear attraction terms ($\langle \mathbf{ab} \rangle$ denotes the trace of the matrix product $\mathbf{ab}$). As usual, the molecular spin-orbitals are expanded in atom centered basis functions ($\sigma = \alpha, \beta$):

$$\psi^{\sigma}_{p}\left(\mathbf{r}\right) = \sum_{\mu} c^{\sigma}_{\mu p} \varphi_{\mu}\left(\mathbf{r}\right) \tag{3.32}$$

with MO coefficients $c^{\sigma}_{\mu p}$. The total density is given by (real orbitals are assumed throughout):

$$\rho\left(\mathbf{r}\right)=\sum_{i\sigma}\left|\psi^{\sigma}_{i}\left(\mathbf{r}\right)\right|^{2}=\sum_{\mu\nu\sigma}P^{\sigma}_{\mu\nu}\varphi_{\mu}\left(\mathbf{r}\right)\varphi_{\nu}\left(\mathbf{r}\right)=\rho^{\alpha}\left(\mathbf{r}\right)+\rho^{\beta}\left(\mathbf{r}\right) \tag{3.33}$$

Where $\mathbf{P}=\mathbf{P}^{\alpha}+\mathbf{P}^{\beta}$ and $P^{\sigma}_{\mu\nu}=\sum_{i_{\sigma}}c^{\sigma}_{\mu i}c^{\sigma}_{\nu i}$.

The second term of (3.31) represents the Coulombic self-repulsion. The third term represents the contribution of the Hartree-Fock exchange with the two-electron integrals being defined as:

$$\left(\mu\nu|\kappa\tau\right)=\int\int\phi_{\mu}\left(\mathbf{r}_{1}\right)\phi_{\nu}\left(\mathbf{r}_{1}\right)r^{-1}_{12}\phi_{\kappa}\left(\mathbf{r}_{2}\right)\phi_{\tau}\left(\mathbf{r}_{2}\right)d\mathbf{r}_{1}d\mathbf{r}_{2} \tag{3.34}$$

The mixing parameter a_x controls the fraction of Hartree-Fock exchange and is of a semi-empirical nature. The exchange correlation contribution may be written as:

$$E_{\mathrm{XC}}\left[\rho_{\alpha},\rho_{\beta}\right]=\left(1-a_{x}\right)E^{\mathrm{GGA}}_{\mathrm{X}}\left[\rho_{\alpha},\rho_{\beta}\right]+bE^{\mathrm{GGA}}_{\mathrm{C}}\left[\rho_{\alpha},\rho_{\beta}\right] \tag{3.35}$$

Here, $E^{\mathrm{GGA}}_{\mathrm{X}}\left[\rho_{\alpha},\rho_{\beta}\right]$ is the exchange part of the XC- functional in question and $E^{\mathrm{GGA}}_{\mathrm{C}}\left[\rho_{\alpha},\rho_{\beta}\right]$ is the correlation part. The parameter b controls the mixing of DFT correlation into the total energy and the parameter c_{DF} is a global scaling factor that allows one to proceed from Hartree-Fock theory ($a_{\mathrm{X}}=1$, $c_{\mathrm{DF}}=0$, $c_{\mathrm{PT}}=0$) to MP2 theory ($a_{\mathrm{X}}=1$, $c_{\mathrm{DF}}=0$, $c_{\mathrm{PT}}=1$) to pure DFT ($a_{\mathrm{X}}=1$, $c_{\mathrm{DF}}=0$, $c_{\mathrm{PT}}=1$) to hybrid DFT ($0<a_{\mathrm{X}}<1$, $c_{\mathrm{DF}}=1$, $c_{\mathrm{PT}}=0$) and finally to the general perturbatively corrected methods discussed in this work ($0<a_{\mathrm{X}}<1$, $c_{\mathrm{DF}}=1$, $0<c_{\mathrm{PT}}<1$). As discussed in detail by Grimme, the B2- PLYP functional uses the Lee-Yang-Parr (LYP) functional as correlation part, the Becke 1988 (B88) functional as GGA exchange part and the optimum choice of the semi-empirical parameters was determined to be $a_{\mathrm{X}}=0.53$, $c_{\mathrm{PT}}=0.27$, $c_{\mathrm{DF}}=1$, $b=1-c_{\mathrm{PT}}$. For convenience, we will suppress the explicit reference to the parameters a_{X} and b in the XC part and rewrite the gradient corrected XC energy as:

$$E_{\mathrm{XC}}\left[\rho^{\alpha},\rho^{\beta}\right]=\int f\left(\rho^{\alpha},\rho^{\beta},\gamma^{\alpha\alpha},\gamma^{\beta\beta},\gamma^{\alpha\beta}\right)d\mathbf{r} \tag{3.36}$$

with the gradient invariants $\gamma^{\sigma\sigma'}=\vec{\nabla}\rho^{\sigma}\vec{\nabla}\rho^{\sigma'}$. The final term in eq (48) represents the scaled second order perturbation energy:

$$E^{\mathrm{PT2}}=\frac{1}{2}\sum_{i_{\alpha}<j_{\alpha}}\left\langle\mathbf{t}^{i_{\alpha}j_{\alpha}}\mathbf{K}^{i_{\alpha}j_{\alpha}+}\right\rangle+\frac{1}{2}\sum_{i_{\beta}<j_{\beta}}\left\langle\mathbf{t}^{i_{\beta}j_{\beta}}\mathbf{K}^{i_{\beta}j_{\beta}+}\right\rangle+\sum_{i_{\alpha},j_{\beta}}\left\langle\mathbf{t}^{i_{\alpha}j_{\beta}}\mathbf{K}^{i_{\alpha}j_{\beta}+}\right\rangle \tag{3.37}$$

The PT2 amplitudes have been collected in matrices $\mathbf{t}^{i_{\sigma}j_{\sigma'}}$ with elements:

$$t^{i_{\sigma}j_{\sigma'}}_{a_{\sigma}b_{\sigma'}}=\bar{K}^{i_{\sigma}j_{\sigma'}}_{a_{\sigma}b_{\sigma'}}\left(\varepsilon^{\sigma}_{i}+\varepsilon^{\sigma'}_{j}-\varepsilon^{\sigma}_{a}-\varepsilon^{\sigma'}_{b}\right)^{-1} \tag{3.38}$$

Where the orbitals were assumed to be canonical with orbital energies ε^{σ}_{p}. The exchange operator matrices are $K^{i_{\sigma}j_{\sigma'}}_{a_{\sigma}b_{\sigma'}}=\left(i_{\sigma}a_{\sigma}|j_{\sigma'}b_{\sigma'}\right)$ and the anti-symmetrized exchange integrals are defined as $\bar{K}^{i_{\sigma}j_{\sigma'}}_{a_{\sigma}b_{\sigma'}}=\left(i_{\sigma}a_{\sigma}|j_{\sigma'}b_{\sigma'}\right)-\delta_{\sigma\sigma'}\left(i_{\sigma}b_{\sigma}|_{\sigma}a_{\sigma}\right)$.

The orbitals satisfy the SCF equations with the matrix element of the SCF operator given by:

$$F^{\sigma}_{\mu\nu}=h_{\mu\nu}+\sum_{\kappa\tau}P_{\kappa\tau}\left(\mu\nu|\kappa\tau\right)-a_{\mathrm{X}}P^{\sigma}_{\kappa\tau}\left(\mu\kappa|\nu\tau\right)+c_{\mathrm{DF}}\left(\mu|V^{\sigma}_{\mathrm{XC}}|\nu\right) \tag{3.39}$$

The matrix elements of the XC–potential for a gradient corrected functional are: [180]

$$\left(\mu|V^{\alpha}_{\mathrm{XC}}|\nu\right)=\int\left\{\frac{\delta f}{\delta\rho_{\alpha}\left(\mathbf{r}\right)}\left(\varphi_{\mu}\varphi_{\nu}\right)+2\frac{\delta f}{\delta\gamma_{\alpha\alpha}}\vec{\nabla}\rho_{\alpha}\vec{\nabla}\left(\varphi_{\mu}\varphi_{\nu}\right)+\frac{\delta f}{\delta\gamma_{\alpha\beta}}\vec{\nabla}\rho_{\beta}\vec{\nabla}\left(\varphi_{\mu}\varphi_{\nu}\right)\right\}d\mathbf{r} \tag{3.40}$$

The energy in equation (3.31) depends on the MO-coefficients, the PT2-amplitudes and through V_{NN}, V_{eN} (in h) and the basis functions also explicitly on the molecular geometry. Unfortunately, the energy is only stationary with respect to the PT2 amplitudes since they can be considered as having been optimized through the minimization of the Hylleraas functional:

$$E_{\mathrm{PT2}}=\min_{\mathbf{t}}\left\{\frac{1}{2}\sum_{i_{\alpha}<j_{\alpha}}\left\langle\mathbf{t}^{i_{\alpha}j_{\alpha}}\mathbf{K}^{i_{\alpha}j_{\alpha}+}\right\rangle+\frac{1}{2}\sum_{i_{\beta}<j_{\beta}}\left\langle\mathbf{t}^{i_{\beta}j_{\beta}}\mathbf{K}^{i_{\beta}j_{\beta}+}\right\rangle+\sum_{i_{\alpha}j_{\beta}}\left\langle\mathbf{t}^{i_{\alpha}j_{\beta}}\mathbf{K}^{i_{\alpha}j_{\beta}+}\right\rangle+\left\langle\mathbf{D'}^{\alpha}\mathbf{F}^{\alpha+}\right\rangle+\left\langle\mathbf{D'}^{\beta}\mathbf{F}^{\beta+}\right\rangle\right\} \tag{3.41}$$

The unrelaxed PT2 difference density is defined as:

$$D'^{\alpha}_{ij} = -\frac{1}{2}\sum_{k_\alpha}\left\langle \mathbf{t}^{i_\alpha k_\alpha}\mathbf{t}^{k_\alpha j_\alpha}\right\rangle - \sum_{k_\beta}\left\langle \mathbf{t}^{i_\alpha k_\beta}\mathbf{t}^{k_\beta j_\alpha}\right\rangle \tag{3.42}$$

$$D'^{\alpha}_{ab} = \sum_{i_\alpha < j_\alpha}\mathbf{t}^{i_\alpha j_\alpha}\mathbf{t}^{i_\alpha j_\alpha +} + \sum_{i_\beta j_\alpha}\mathbf{t}^{i_\beta j_\alpha +}\mathbf{t}^{i_\beta j_\alpha} \tag{3.43}$$

With analogous expressions for the spin-down unrelaxed difference densities. Minimization of this functional with respect to the amplitudes yields the second order perturbation energy. The derivative of the SCF part of equation (3.31) with respect to a parameter "x" is straightforward and well known. It yields:

$$\begin{aligned} E^x_{\text{SCF}} = V^x_{NN} + \langle \mathbf{P}\mathbf{h}^x\rangle + \left\langle \mathbf{W}^{SCF}\mathbf{S}^{(x)}\right\rangle + \textstyle\sum_{\mu\nu\kappa\tau}\Gamma_{\mu\nu\kappa\tau}\left(\mu\nu|\kappa\tau\right)^{(x)} \\ + \underbrace{\sum_{\sigma}}_{(\sigma'\neq\sigma)} \int\left\{\tfrac{\delta f}{\delta\rho_\sigma(\mathbf{r})}\rho^{(x)}_\sigma + 2\tfrac{\delta f}{\delta\gamma_{\sigma\sigma}}\vec{\nabla}\rho_\sigma\vec{\nabla}\rho^{(x)}_\sigma + \tfrac{\delta f}{\delta\gamma_{\sigma\sigma'}}\vec{\nabla}\rho_{\sigma'}\vec{\nabla}\rho^{(x)}_\sigma\right\} d\mathbf{r} \end{aligned} \tag{3.44}$$

Superscript "x" refers to the derivative with respect to some perturbation "x" while a superscript in parentheses indicates that only the derivative of the basis functions with respect to "x" is to be taken. For example:

$$\begin{aligned} \rho^{(x)}_\sigma &= \textstyle\sum_{\mu\nu}P^\sigma_{\mu\nu}\left\{\tfrac{\partial\varphi_\mu}{\partial x}\varphi_\nu + \varphi_\mu\tfrac{\partial\varphi_\nu}{\partial x}\right\} \\ h^x_{\mu\nu} &= \left(\tfrac{\partial\varphi_\mu}{\partial x}|\hat{h}|\varphi_\nu\right) + \left(\varphi_\mu|\hat{h}|\tfrac{\partial\varphi_\nu}{\partial x}\right) + \left(\varphi_\mu|\tfrac{\partial\hat{h}}{\partial x}|\varphi_\nu\right) \end{aligned} \tag{3.45}$$

In equation (3.44), $\mathbf{S}$ is the overlap matrix and $\mathbf{W}^{\text{SCF}}$ the energy weighted density:

$$W^{\text{SCF}}_{\mu\nu} = W^{\alpha;\text{SCF}}_{\mu\nu} + W^{\beta;\text{SCF}}_{\mu\nu} = -\sum_{i\sigma}c^\sigma_{\mu i}c^\sigma_{\nu i}\varepsilon^\sigma_i \tag{3.46}$$

At this point, the effective two-particle density matrix is fully separable and reads:

$$\Gamma_{\mu\nu\kappa\tau} = \frac{1}{2}P_{\mu\nu}P_{\kappa\tau} - \frac{1}{2}a_xP^\alpha_{\mu\kappa}P^\alpha_{\nu\tau} - \frac{1}{2}a_xP^\beta_{\mu\kappa}P^\beta_{\nu\tau} \tag{3.47}$$

The derivative of the PT2 part is considerably more complex, since E_{PT2} is not stationary with respect to changes in the molecular orbitals. This necessitates the solution of the coupled-perturbed SCF (CP-SCF) equations. We follow the standard practice and expand the perturbed orbitals in terms of the unperturbed ones as:

$$\psi^{\sigma;x}_p(\mathbf{r}) = \sum_q U^{\sigma;x}_{qp}\psi^\sigma_q(\mathbf{r}) \tag{3.48}$$

The occupied-occupied and virtual-virtual blocks of $\mathbf{U}$ are fixed, as usual, through the derivative of the orthonormality constraints:

$$U^{\sigma;x}_{ij} = -\frac{1}{2}S^{\sigma(x)}_{ij} \tag{3.49}$$

$$U^{\sigma;x}_{ab} = -\frac{1}{2}S^{\sigma(x)}_{ab} \tag{3.50}$$

$$U^{\sigma;x}_{ia} = -S^{\sigma(x)}_{ia} - U^{\sigma;x}_{ai} \tag{3.51}$$

Where $S^{\sigma(x)}_{pq} = \sum_{\mu\nu}c^\sigma_{\mu p}c^\sigma_{\nu q}S^{(x)}_{\mu\nu}$. The remaining virtual-occupied block of $\mathbf{U}^{\,x}$ must be determined through the solution of the CP-SCF equations. However, as shown by Handy and Schaefer, this step is unnecessary and only a single set of CP-SCF equations (Z-vector equations) needs to be solved. To this end, one defines the Lagrangian:

$$\begin{aligned} L^\alpha_{ai} = R^\sigma(\mathbf{D}')_{ai} + 2\textstyle\sum_{j_\alpha b_\alpha c_\alpha}\left(a_\alpha c_\alpha|j_\alpha b_\alpha\right)t^{i_\alpha j_\alpha}_{c_\alpha b_\alpha} - 2\sum_{j_\alpha k_\alpha b_\alpha}\left(k_\alpha i_\alpha|j_\alpha b_\alpha\right)t^{k_\alpha j_\alpha}_{a_\alpha b_\alpha} \\ +2\textstyle\sum_{j_\beta b_\beta c_\alpha}\left(a_\alpha c_\alpha|j_\beta b_\beta\right)t^{j_\beta i_\alpha}_{b_\beta c_\alpha} - 2\sum_{j_\beta k_\alpha b_\beta}\left(k_\alpha i_\alpha|j_\beta b_\beta\right)t^{j_\beta k_\alpha}_{b_\beta a_\alpha} \end{aligned} \tag{3.52}$$

An analogous equation holds for L^β_{ai}. The matrix elements of the response operator $R^\alpha(\mathbf{D}')$ are best evaluated in the AO basis and then transformed into the MO basis. The AO basis matrix elements are given by:

$$\begin{aligned} R^\alpha(\mathbf{D}')_{\mu\nu} = &\sum_{\kappa\tau}2D'_{\kappa\tau}\left(\mu\nu|\kappa\tau\right) - D'^{\alpha}_{\kappa\tau}\left[(\mu\kappa|\nu\tau) + (\nu k|\mu\tau)\right] \\ &+\sum_\zeta\int\left[\frac{\delta^2 f}{\delta\rho_\alpha\delta\zeta}\zeta(\mathbf{D}')(\phi_\mu\phi_\nu) + \left(2\frac{\delta^2 f}{\delta\gamma_{\alpha\alpha}\delta\zeta}\vec{\nabla}\rho^\alpha_{\mathbf{P}} + \frac{\delta^2 f}{\delta\gamma_{\alpha\beta}\delta\zeta}\vec{\nabla}\rho^\beta_{\mathbf{P}}\right)\zeta(\mathbf{D}')\vec{\nabla}(\phi_\mu\phi_\nu)\right. \\ &\left.+\left(2\frac{\delta f}{\delta\gamma_{\alpha\alpha}}\vec{\nabla}\rho^\alpha_{\mathbf{D}'} + \frac{\delta f}{\delta\gamma_{\alpha\beta}}\vec{\nabla}\rho^\beta_{\mathbf{D}'}\right)\vec{\nabla}(\phi_\mu\phi_\nu)\right]d\mathbf{r} \end{aligned} \tag{3.53}$$

where

$$\zeta\left(\mathbf{D}'\right)=\rho_{\mathbf{D}'}^{\alpha},\rho_{\mathbf{D}'}^{\beta},\gamma_{\alpha\alpha}\left(\mathbf{D}'\right),\gamma_{\beta\beta}\left(\mathbf{D}'\right),\gamma_{\alpha\beta}\left(\mathbf{D}'\right) \tag{3.54}$$

The ζ-gradient-parameters are evaluated as a mixture of PT2 difference densities and SCF densities. For example:

$$\gamma_{\alpha\alpha}(\mathrm{D}')=2\vec{\nabla}\rho_{\mathrm{D}'}^{\alpha}\vec{\nabla}\rho_{\mathrm{P}'}^{\alpha} \tag{3.55}$$

With

$$\rho_{\mathbf{D}'}^{\alpha}\left(\mathbf{r}\right)=\sum_{\mu\nu}D_{\mu\nu}^{'\alpha}\phi_{\mu}\left(\mathbf{r}\right)\phi_{\nu}\left(\mathbf{r}\right) \tag{3.56}$$

$$\rho_{\mathbf{P}}^{\alpha}\left(\mathbf{r}\right)=\sum_{\mu\nu}P_{\mu\nu}^{\alpha}\phi_{\mu}\left(\mathbf{r}\right)\phi_{\nu}\left(\mathbf{r}\right) \tag{3.57}$$

Having defined the Lagrangian, the following CP-SCF equations need to be solved for the elements of the "Z- vector":

$$\left(\varepsilon_{a}^{\sigma}-\varepsilon_{i}^{\sigma}\right)Z_{ai}^{\sigma}+R^{\sigma}\left(\mathbf{Z}\right)_{ai}=-L_{ai}^{\sigma} \tag{3.58}$$

The solution defines the occupied-virtual block of the relaxed difference density, which is given by:

$$\mathbf{D}^{\sigma}=\mathbf{D}'^{\sigma}+\mathbf{Z}^{\sigma} \tag{3.59}$$

For convenience, $\mathbf{D}^{\sigma}$ is symmetrized since it will only be contracted with symmetric matrices afterwards. After having solved the Z-vector equations, all parts of the energy weighted difference density matrix can be readily calculated:

$$W_{ij}^{\alpha;\mathrm{PT2}}=-\frac{1}{2}D_{ij}^{\alpha}\left(\varepsilon_{i}^{\alpha}+\varepsilon_{j}^{\alpha}\right)-\frac{1}{2}R\left(\mathbf{D}\right)_{ij}-\sum_{k_{\alpha}a_{\alpha}b_{\alpha}}\left(i_{\alpha}a_{\alpha}|k_{\alpha}b_{\alpha}\right)t_{a_{\alpha}b_{\alpha}}^{j_{\alpha}k_{\alpha}}-\sum_{k_{\beta}a_{\alpha}b_{\beta}}\left(i_{\alpha}a_{\alpha}|k_{\beta}b_{\beta}\right)t_{b_{\beta}a_{\alpha}}^{k_{\beta}j_{\alpha}} \tag{3.60}$$

$$W_{ab}^{\alpha;\mathrm{PT2}}=-\frac{1}{2}D_{ab}^{\alpha}\left(\varepsilon_{a}^{\alpha}+\varepsilon_{b}^{\alpha}\right)-\sum_{i_{\alpha}j_{\alpha}c_{\alpha}}\left(i_{\alpha}a_{\alpha}|j_{\alpha}c_{\alpha}\right)t_{b_{\alpha}c_{\alpha}}^{i_{\alpha}j_{\alpha}}-\sum_{i_{\alpha}j_{\beta}c_{\beta}}\left(i_{\alpha}a_{\alpha}|j_{\beta}c_{\beta}\right)t_{c_{\beta}b_{\alpha}}^{j_{\beta}i_{\alpha}} \tag{3.61}$$

$$W_{ai}^{\alpha;\mathrm{PT2}}=-2\sum_{j_{\alpha}k_{\alpha}b_{\alpha}}\left(k_{\alpha}i_{\alpha}|j_{\alpha}b_{\alpha}\right)t_{a_{\alpha}b_{\alpha}}^{k_{\alpha}j_{\alpha}}-2\sum_{j_{\beta}k_{\alpha}b_{\beta}}\left(k_{\alpha}i_{\alpha}|j_{\beta}b_{\beta}\right)t_{b_{\beta}a_{\alpha}}^{j_{\beta}k_{\alpha}} \tag{3.62}$$

$$W_{ia}^{\alpha;\mathrm{PT2}}=-\varepsilon_{i}^{\alpha}Z_{ai}^{\alpha} \tag{3.63}$$

Once more, analogous equations hold for the spin-down case. With the relaxed difference density and energy weighted density matrices in hand, one can finally proceed to evaluate the gradient of the PT2 part as ($\mathbf{W}^{\mathrm{PT2}}=\mathbf{W}^{\alpha;\mathrm{PT2}}+\mathbf{W}^{\beta;\mathrm{PT2}}$):

$$\begin{aligned}E_{\mathrm{PT2}}^{x}&=\left\langle\mathbf{D}\mathbf{h}^{x}\right\rangle+\left\langle\mathbf{W}^{\mathrm{PT2}}\mathbf{S}^{(x)}\right\rangle+\textstyle\sum_{\mu\nu\kappa\tau}\Gamma_{\mu\nu\kappa\tau}^{\mathrm{PT2}}\left(\mu\nu|\kappa\tau\right)^{(x)}\\&+\textstyle\sum_{\substack{\sigma\\(\sigma\neq\sigma')}}\int\left\{\frac{\delta f}{\delta\rho_{\sigma}(\mathbf{r})}\rho_{\sigma}^{(x)}+2\frac{\delta f}{\delta\gamma_{\sigma\sigma}}\overset{r}{\nabla}\rho_{\sigma}\overset{r}{\nabla}\rho_{\sigma}^{(x)}+\frac{\delta f}{\delta\gamma_{\sigma\sigma}}\overset{r}{\nabla}\rho_{\sigma'}\overset{r}{\nabla}\rho_{\sigma}^{(x)}\right\}d\mathbf{r}\end{aligned} \tag{3.64}$$

The final derivative of eq. (3.31) is of course the sum $E_{SCF}^{x}+c_{PT}E_{\mathrm{PT2}}^{x}$. Both derivatives should be evaluated simultaneously in the interest of computational efficiency.

Note that the exchange-correlation contributions to the gradient take a somewhat more involved form than might have been anticipated. In fact, from looking at the SCF XC-gradient (eq. (3.44)) it could have been speculated that the PT2 part of the gradient is of the same form but with $\rho_{\mathbf{P}}^{\sigma(x)}$ being replaced by $\hat{H}$, the relaxed PT2 difference density. This is, however, not the case. The underlying reason for the added complexity apparent in equation (3.64) is that the XC contributions to the PT2 gradient arise from the contraction of the relaxed PT2 difference density with the derivative of the SCF operator. Since the SCF operator already contains the first derivative of the XC potential and the PT2 energy is not stationary with respect to changes in the SCF density, a response type term arises which requires the evaluation of the second functional derivative of the XC-functional. Finally, as is well known from MP2 gradient theory, the effective two- particle density matrix contains a separable and a non-separable part:

$$\Gamma_{\mu\nu\kappa\tau}^{\mathrm{PT2}}=D_{\mu\nu}P_{\kappa\tau}-D_{\mu\kappa}^{\alpha}P_{\nu\tau}^{\alpha}-D_{\mu\kappa}^{\beta}P_{\nu\tau}^{\beta}+\Gamma_{\mu\nu\kappa\tau}^{\mathrm{NS}} \tag{3.65}$$

$$\Gamma_{\mu\nu\kappa\tau}^{\mathrm{NS}}=\sum_{i_{\alpha}j_{\alpha}a_{\alpha}b_{\alpha}}c_{\mu i}^{\alpha}c_{\nu a}^{\alpha}c_{\kappa j}^{\alpha}c_{\tau b}^{\alpha}t_{a_{\alpha}b_{\alpha}}^{i_{\alpha}j_{\alpha}}+\sum_{i_{\beta}j_{\beta}a_{\beta}b_{\beta}}c_{\mu i}^{\beta}c_{\nu a}^{\beta}c_{\kappa j}^{\beta}c_{\tau b}^{\beta}t_{a_{\beta}b_{\beta}}^{i_{\beta}j_{\beta}}+2\sum_{i_{\alpha}j_{\beta}a_{\alpha}b_{\beta}}c_{\mu i}^{\alpha}c_{\nu a}^{\alpha}c_{\kappa j}^{\beta}c_{\tau b}^{\beta}t_{a_{\alpha}b_{\beta}}^{i_{\alpha}j_{\beta}} \tag{3.66}$$

Thus, the non-separable part is merely the back-transformation of the amplitudes from the MO to the AO basis. It is, however, important to symmetrize the two-particle density matrix in order to be able to exploit the full permutational symmetry of the AO derivative integrals.

3.9.14 Orbital Optimized MP2 Methods

OO-MP2 Theory

The MP2 energy can be regarded as being stationary with respect to the MP2 amplitudes, since they can be considered as having been optimized through the minimization of the Hylleraas functional:

$$E_{\text{MP2}} = \min_{\mathbf{t}} \left\{ 2 \left\langle \Psi_1 | \hat{H} | \Psi_0 \right\rangle + \left\langle \Psi_1 | \hat{H}_0 - E_0 | \Psi_1 \right\rangle \right\} \tag{3.67}$$

$\hat{H}$ is the 0th order Hamiltonian as proposed by Møller and Plesset, Ψ_0 is the reference determinant, Ψ_1 is the first-order wave function and $E_0 = E_{\text{HF}} = \langle \Psi_{\text{HF}} | \hat{H} | \Psi_{\text{HF}} \rangle$ is the reference energy. The quantities **t** collectively denote the MP2 amplitudes.

The fundamental idea of the OO-MP2 method is to not only minimize the MP2 energy with respect to the MP2 amplitudes, but to minimize the total energy additionally with respect to changes in the orbitals. Since the MP2 energy is not variational with respect to the MO coefficients, no orbital relaxation due to the correlation field is taken into account. If the reference determinant is poor, the low-order perturbative correction then becomes unreliable. This may be alleviated to a large extent by choosing better orbitals in the reference determinant. Numerical evidence for the correctness of this assumption will be presented below.

In order to allow for orbital relaxation, the Hylleraas functional can be regarded as a functional of the wavefunction amplitudes **t** and the orbital rotation parameters **R** that will be defined below. Through a suitable parameterization it becomes unnecessary to ensure orbital orthonormality through Lagrange multipliers. The functional that we minimize reads:

$$L\left\{\mathbf{t}, \mathbf{R}\right\} = E_0\left[\mathbf{R}\right] + 2\left\langle \Psi_1 \right| \hat{H} \left| \Psi_0 \right\rangle + \left\langle \Psi_1 \right| \hat{H}_0 - E_0 \left| \Psi_1 \right\rangle \tag{3.68}$$

Ψ_0 is the reference determinant. However, it does no longer correspond to the Hartree-Fock (HF) determinant. Hence, the reference energy $E_0\left[\mathbf{R}\right] = \left\langle \Psi_0\left[\mathbf{R}\right] \right| \hat{H} \left| \Psi_0\left[\mathbf{R}\right] \right\rangle$ also changes during the variational process and is no longer stationary with respect to the HF MO coefficients. Obviously, $E_0\left[\mathbf{R}\right] \geqslant E_{\text{HF}}$ since the HF determinant is, by construction, the single determinant with the lowest expectation value of the full Hamiltonian.

The reference energy is given as:

$$E_0\left[\mathbf{R}\right] = \sum_i \left\langle i \right| h \left| i \right\rangle + \frac{1}{2} \sum_{ij} \left\langle ij || ij \right\rangle \tag{3.69}$$

The first-order wave function excluding single excitations is:

$$\left| \Psi_1 \right\rangle = \frac{1}{4} \sum_{ijab} t_{ab}^{ij} \left| \Psi_{ij}^{ab} \right\rangle \tag{3.70}$$

A conceptually important point is that Brillouin's theorem [356] is no longer obeyed since the Fock matrix will contain off-diagonal blocks. Under these circumstances the first-order wavefunction would contain contributions from single excitations. Since the orbital optimization brings in all important effects of the singles we prefer to leave them out of the treatment. Any attempt to the contrary will destroy the convergence properties. We have nevertheless contemplated to include the single excitations perturbatively:

$$E_{\text{Singles}}^{(2)} = -\sum_{ia} \frac{\left| F_{ia} \right|^2}{\varepsilon_a - \varepsilon_i} \tag{3.71}$$

The perturbative nature of this correction would destroy the stationary nature of the total energy and is hence not desirable. Furthermore, results with inclusion of single excitation contributions represent no improvement to the results reported below. They will therefore not be documented below and henceforth be omitted from the OO-MP2 method by default.

The explicit form of the orbital-optimized MP2 Hylleraas functional employing the RI approximation (OO-RI-MP2) becomes:

$$L_\infty\left[\mathbf{t}, \mathbf{R}\right] = \sum_i \left\langle i \right| \hat{h} \left| i \right\rangle + \frac{1}{2} \sum_{ij} \left\langle ij || ij \right\rangle + \sum_{iaP} (ia|P) \Gamma_{ia}^{\prime P} + \sum_{ij} D_{ij} F_{ij} + \sum_{ab} D_{ab} F_{ab} \tag{3.72}$$

with:

$$\Gamma'^{P}_{ia} = \sum_{Q} V^{-1}_{PQ} \sum_{jb} (Q|jb) t^{ij}_{ab} \tag{3.73}$$

$$(ia|P) = \int\int \psi_i(\mathbf{r}_1)\psi_a(\mathbf{r}_1)\frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|}\eta_P(\mathbf{r}_2)d\mathbf{r}_1 d\mathbf{r}_2 \tag{3.74}$$

$$(P|Q) = \int\int \eta_p(\mathbf{r}_1)\frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|}\eta_Q(\mathbf{r}_2)d\mathbf{r}_1 d\mathbf{r}_2 \tag{3.75}$$

Here, $\{\psi\}$ is the set of orthonormal molecular orbitals and $\{\eta\}$ denotes the auxiliary basis set. F_{pq} denotes a Fock matrix element:

$$F_{pq} = \langle p| \hat{h} |q\rangle + \sum_{k} \langle pk||qk\rangle \tag{3.76}$$

and it is insisted that the orbitals diagonalize the occupied and virtual subspaces, respectively:

$$\begin{aligned} F_{ij} &= \delta_{ij} F_{ii} = \delta_{ij}\varepsilon_i \\ F_{ab} &= \delta_{ab} F_{aa} = \delta_{ab}\varepsilon_a \end{aligned} \tag{3.77}$$

The MP2 like density blocks are,

$$\begin{aligned} D_{ij} &= -\tfrac{1}{2}\textstyle\sum_{kab} t^{ik}_{ab} t^{jk}_{ab} \\ D_{ab} &= \tfrac{1}{2}\textstyle\sum_{ijc} t^{ij}_{ac} t^{ij}_{bc} \end{aligned} \tag{3.78}$$

where the MP2 amplitudes in the case of a block diagonal Fock matrix are obtained through the condition $\frac{\partial L_\infty}{\partial t^{ij}_{ab}} = 0$:

$$t^{ij}_{ab} = -\frac{\langle ij||ab\rangle}{\varepsilon_a + \varepsilon_b - \varepsilon_i - \varepsilon_j} \tag{3.79}$$

The orbital changes are parameterized by an anti-Hermitian matrix **R** and an exponential Ansatz,

$$\begin{aligned} \mathbf{c}^{\text{new}} &= \mathbf{c}^{\text{old}} \exp(\mathbf{R}) \\ \mathbf{R} &= \begin{pmatrix} 0 & \mathbf{R}_{ia} \\ -\mathbf{R}_{ia} & 0 \end{pmatrix} \end{aligned} \tag{3.80}$$

The orbitals changes to second order are,

$$\begin{aligned} \exp(\mathbf{R})\,|i\rangle &= |i\rangle + \textstyle\sum_a \mathbf{R}_{ai}\,|a\rangle - \tfrac{1}{2}\sum_{jb} \mathbf{R}_{bi}\mathbf{R}_{bj}\,|j\rangle + \ldots \\ \exp(\mathbf{R})\,|a\rangle &= |a\rangle - \textstyle\sum_i \mathbf{R}_{ai}\,|i\rangle - \tfrac{1}{2}\sum_{jb} \mathbf{R}_{aj}\mathbf{R}_{bj}\,|b\rangle + \ldots \end{aligned} \tag{3.81}$$

Through this Ansatz it is ensured that the orbitals remain orthonormal and no Lagrangian multipliers need to be introduced. The first-order expansion of the Fock operator due to the orbital rotations are:

$$F_{pq}[R] = F_{pq}[0] + R^{(1)}_{pq} + \sum_{r} R_{rp}F_{rq}[0] + R_{rq}F_{pr}[0] \tag{3.82}$$

$$R^{(1)}_{pq} = \sum_{kc} R_{ck}\left\{\langle pc||qk\rangle + \langle pk||qc\rangle\right\} \tag{3.83}$$

The first-order energy change becomes $\left(h_{pq} \equiv \langle p|\hat{h}|q\rangle, g_{pqrs} \equiv \langle pq||rs\rangle\right)$:

$$\begin{aligned} L_\infty[\mathbf{t},\mathbf{R}] = {} & \textstyle\sum_{ic} R_{ci}(h_{ci} + h_{ic}) + \tfrac{1}{2}\sum_{ijc} R_{ci}(g_{cjij} + g_{ijcj}) + R_{cj}(g_{icij} + g_{ijic}) \\ & +2\textstyle\sum_{iacP} R_{ci}(ac|P)\Gamma'^{P}_{ia} - 2\sum_{ikaP} R_{ak}(ik|P)\Gamma'^{P}_{ia} \\ & -\textstyle\sum_{ij} D_{ij}\left(R^{(1)}_{ij} + \sum_c (R_{ci}F_{cj} + R_{cj}F_{ic})\right) \\ & +\textstyle\sum_{ab} D_{ab}\left(R^{(1)}_{ab} - \sum_k (R_{ak}F_{kb} + R_{bk}F_{ak})\right) \end{aligned} \tag{3.84}$$

The condition for the energy functional to be stationary with respect to the orbital rotations $\left(\frac{\partial L_\infty[\mathbf{t},\mathbf{R}]}{\partial R_{ai}} = 0\right)$, yields the expression for the orbital gradient and hence the expression for the OO-RI-MP2 Lagrangian.

$$\frac{\partial L_\infty[\mathbf{t},\mathbf{R}]}{\partial R_{ai}} \equiv g_{ai} = 2F_{ai} + 2\sum_{j} D_{ij}F_{aj} - 2\sum_{b} D_{ab}F_{ib} + R^{(1)}(\mathbf{D})_{ai} \tag{3.85}$$

$$+2\sum_{cP}(ac|P)\Gamma'^{P}_{ia} - 2\sum_{kP}(ik|P)\Gamma'^{P}_{ia}$$

The goal of the orbital optimization process is to bring this gradient to zero. There are obviously many ways to achieve this. In our experience, the following simple procedure is essentially satisfactory. We first build a matrix **B** in the current MO basis with the following structure:

$$\begin{aligned} \mathbf{B}_{ij} &= \delta_{ij}\mathbf{F}_{ii} \\ \mathbf{B}_{ab} &= \delta_{ab}(\mathbf{F}_{aa} + ?) \\ \mathbf{B}_{ai} &= \mathbf{B}_{ia} = \mathbf{g}_{ai} \end{aligned} \tag{3.86}$$

where Δ is a level shift parameter. The occupied/occupied and virtual/virtual blocks of this matrix are arbitrary but their definition has a bearing on the convergence properties of the method. The orbital energies of the block diagonalized Fock matrix appear to be a logical choice. If the gradient is zero, the **B**-matrix is diagonal. Hence one obtains an improved set of orbitals by diagonalizing **B**.

In order to accelerate convergence a standard DIIS scheme is used. [333, 357] However, in order to carry out the DIIS extrapolation of the **B**-matrix it is essential that a common basis is used that does not change from iteration to iteration. Since the **B**-matrix itself is defined in the molecular orbitals of the current iteration we choose as a common set of orthonormal orbitals the MOs of the HF calculation. The extrapolation is carried out in this basis and the extrapolated **B**-matrix is transformed back to the current set of MOs prior to diagonalization. Obviously, the same strategy can be used for orbital optimization in any method for which an orbital gradient is available.

For well behaved cases this simple scheme converges in 5-10 iterations. Transition metals and more complicated molecules may require up to 20 iterations and level shifting in order to achieve convergence.

Upon convergence the sum of the matrix **D** and the density of the reference determinant $P_{\mu\nu} = \sum_i c_{\mu i}c_{\nu i}$ form the true one-particle density matrix of the OO-MP2 approach that can be used for property or gradient calculations.

OO-MP2 - Tips, Usage and Examples

By making the Hylleraas functional stationary with respect to the orbital rotations one obtains the orbital-optimized MP2 method that is implemented in ORCA in combination with the RI approximation (OO-RI-MP2). One obtains from these calculations orbitals that are adjusted to the dynamic correlation field at the level of second order many-body perturbation theory. Also, the total energy of the OO-RI-MP2 method is lower than that of the RI-MP2 method itself. One might think of this method as a special form of multiconfigurational SCF theory except for the fact that the Hamiltonian is divided into a 0th order term and a perturbation.

The main benefit of the OO-RI-MP2 method is that it "repairs" the poor Hartree–Fock orbitals to some extent which should be particularly beneficial for systems which suffer from the imbalance in the Hartree-Fock treatment of the Coulomb and the Exchange hole. Based on the experience gained so far, the OO-RI-MP2 method is no better than RI-MP2 itself for the thermochemistry of organic molecules. However, for reactions barriers and radicals the benefits of OO-MP2 over MP2 are substantial. This is particularly true with respect to the spin-component scaled variant of OO-RI-MP2 that is OO-RI-SCS-MP2. Furthermore, the OO-RI-MP2 method substantially reduces the spin contamination in UHF calculations on radicals.

Since every iteration of the OO-MP2 method is as expensive as a RI-MP2 relaxed density calculation, the computational cost is much higher than for RI-MP2 itself. One should estimate about a factor of 10 increase in computational time with respect to the RI-MP2 time of a normal calculation. This may still be feasible for calculations in the range of 1000–2000 basis functions (the upper limit, however, implies very significant computational costs). A full assessment of the orbital optimized MP2 method has been published.[358]

OO-RI-MP2 is triggered either with `! OO-RI-MP2` or `! OO-RI-SCS-MP2` (with spin component scaling) in the simple input line or by `OrbOpt true` in the `%mp2` block. The method comes with the following new variables:

```
%mp2 OrbOpt  true   # turns on the orbital optimization
     CalcS2  false  # calculate the S**2 expectation value
                    # in spin-unrestricted calculations
     MaxOrbIter 64  # Max. number of iterations
     MP2Shift  0.1  # Level shift for the procedure
     end
```

The solver is a simple DIIS type scheme with additional level shifting. We have found that it is not really beneficial to first converge the Hartree-Fock equations. Thus it is sensible to additionally use the keyword `! noiter` in order to turn off the standard Hartree-Fock SCF process before entering the orbital optimizations.

The OO-RI-MP2 method is implemented for RHF and UHF reference wavefunctions. Analytic gradients are available.

The density does not need to be requested separately in OO-RI-MP2 calculations because it is automatically calculated. Also, there is no distinction between relaxed and unrelaxed densities because the OO-RI-MP2 energy is fully stationary with respect to all wavefunction parameters and hence the unrelaxed and relaxed densities coincide.

3.9.15 Regularized MP2 and RI-MP2

Regularized MP2 is a variant of second-order Moller-Plesset theory (MP2) introduced by J. Shee, M. Loipersberger, A. Rettig, J. Lee, and M. Head-Gordon [359] that aims to improve its accuracy for systems with π-driven dispersion interactions and dative bonds in transition metal complexes. The approach achieves this by introducing a single-parameter, energy-gap dependent regularization that dampens overestimated pairwise additive contributions, thus renormalizing first-order amplitudes to empirically mimic higher-order correlations.

For this, the standard MP2 energy and thus the standard algorithms are modified. For the σ ($p = 1$) and σ^2 ($p = 2$) regularization, the energy is modified according to

$$E_{\sigma^p\text{-MP2}} = -\frac{1}{4}\sum_{ijab}\frac{|\langle ij||ab\rangle|^2}{\Delta_{ij}^{ab}}(1 - e^{-\sigma(\Delta_{ij}^{ab})^p})$$

which corresponds to regularizing the first-order amplitudes. For the κ regularization, the MP2 energy is modified according to

$$E_{\kappa\text{-MP2}} = -\frac{1}{4}\sum_{ijab}\frac{|\langle ij||ab\rangle|^2}{\Delta_{ij}^{ab}}\left(1 - e^{-\kappa(\Delta_{ij}^{ab})}\right)^2$$

which corresponds to regularizing the first-order amplitudes and the exchange integrals.

Regularized MP2 is available for standard MP2 in "memory mode" (Q1Opt>0) and RI-MP2 (RIJDX, RIJCOSX, RIJK).

The usage of regularized MP2 is controlled by the `DoRegMP2` keyword, the type of regularization can be specified by setting the `RegMP2Type` parameter to 0 for κ, 1 for σ, or 2 for σ^2. The value of the regularizers can be specified by `RegMP2Kappa` and `RegMP2Sigma` respectively.

```
%mp2
    DoRegMP2      true    # required
    RegMP2Type    0       # kappa regularizer
                  1       # sigma regularizer
                  2       # sigma-squared regularizer
    RegMP2Kappa   1.1     # kappa value
    RegMP2Sigma   0.5     # sigma value
end
```

It is important to note that only single point energies are available and tested for regularized MP2. Density, Gradient, and Hessian calculations are not yet supported.

3.9.16 Keywords

Table 3.34: Simple input keywords selecting MP2 variants.

Keyword	Description
MP2	Canonical MP2
RI-MP2	RI-MP2 (Aliases: RIMP2, MP2RI)
DLPNO-MP2	DLPNO-MP2
SCS-MP2	Spin-component-scaled canonical MP2
SCS-MP2	Spin-opposite-scaled canonical MP2
RI-SCS-MP2	Spin-component-scaled RI-MP2 (Aliases: SCS-RI-MP2, SCS-RIMP2)
RI-SOS-MP2	Spin-opposite-scaled RI-MP2 (Aliases: SOS-RI-MP2, SOS-RIMP2)
DLPNO-SCS-MI	Spin-component-scaled DLPNO-MP2 (Alias: SCS-DLPNO-MP2)
DLPNO-SOS-MI	Spin-opposite-scaled DLPNO-MP2 (Alias: SOS-DLPNO-MP2)
RI-SCS-MP2	Spin-opposite-scaled RI-MP2 (Aliases: SOS-RI-MP2, SOS-RIMP2)
OO-RI-MP2	Orbital-optimized RI-MP2 (Aliases: OO-RIMP2, OO-MP2RI)
OO-RI-SCS-MI	Orbital-optimized spin-component-scaled RI-MP2 (Aliases: OO-SCS-RI-MP2, OO-SCS-RIMP2)
OO-RI-SOS-MI	Orbital-optimized spin-opposite-scaled RI-MP2 (Aliases: OO-SOS-RI-MP2, OO-SOS-RIMP2)
F12-MP2	MP2 with F12 correction (Alias: MP2-F12)
F12-RI-MP2	RI-MP2 with F12 correction (Aliases: MP2-F12-RI, RI-MP2-F12)
F12/D-RI-MP2	RI-MP2 with F12 correction employing the lower-cost D approximation (Aliases: MP2-F12/D-RI, RI-MP2-F12/D, F12D-RI-MP2, MP2-F12D-RI, RI-MP2-F12D)
F12-DLPNO-MI	DLPNO-MP2 with F12 correction (Alias: DLPNO-MP2-F12)
F12/D-DLPNO-MP2	DLPNO-MP2 with F12 correction employing the lower-cost D approximation (Aliases: DLPNO-MP2-F12/D, F12D-DLPNO-MP2, DLPNO-MP2-F12D)
LoosePNO	Selects loose thresholds for DLPNO-MP2
NormalPNO	(Default) Selects default thresholds for DLPNO-MP2
TightPNO	Selects tight thresholds for DLPNO-MP2

3.10 Coupled Cluster and CI Theories (MDCI)

Aside from the *second-order Møller-Plesset theory (MP2)*, ORCA features a variety of single-reference correlation methods for single point energies (restricted to a RHF or RKS determinant in the closed-shell case and a UHF or UKS determinant in the open-shell case; quasi-restricted orbitals (QROs)[360] are also supported in the open-shell case). These methods are all fairly expensive but maybe be used in order to obtain accurate results in the case that the reference determinant is a good starting point for the expansion of the many-body wavefunction. The methods are implemented in the orca_mdci module, which is the main subject of this section. 'MDCI' is an abbreviations for "matrix driven configuration interaction". The term is rather technical to emphasize that if one wants to implement these methods (CCSD, QCISD etc.) efficiently, one needs to write them in terms of matrix operations, which pretty much every computer can drive at peak performance.

It should be noted that in recent years, a number of highly correlated single reference methods and their properties were added to ORCA using *automatic code generation and reside in the orca_autoci module*, which follows similar design principles.[361, 362] Let us first briefly describe the theoretical background of the methods, that we have implemented in ORCA.

3.10.1 Theory

We start from the full CI hierarchy in which the wavefunction is expanded as:

$$|\Psi\rangle = |0\rangle + |S\rangle + |D\rangle + |T\rangle + |Q\rangle + ... \tag{3.87}$$

where $|0\rangle$ is a single-determinant reference and S, D, T, Q, …denote the single, double, triple quadruple and higher excitations relative to this determinant at the spin-orbital level. As usual, labels i, j, k, l refer to occupied orbitals in $|0\rangle$, a, b, c, d to unoccupied MOs and p, q, r, s to general MOs. The action of the second quantized excitation operators $a_i^a = a_a^\dagger a_i$ on $|0\rangle$ lead to excited determinants $|\Phi_i^a\rangle$ that enter $|\Psi\rangle$ with coefficients C_a^i. The variational equations are:

$$\langle \Phi_i^a | H - E_0 | 0 + S + D\rangle = E_C C_a^i - \langle \Phi_i^a | H - E_0 | T\rangle \tag{3.88}$$

$$\left\langle \Phi_{ij}^{ab} \left| H - E_0 \right| 0 + S + D \right\rangle = E_{\mathrm{C}} C_{ab}^{ij} - \left\langle \Phi_{ij}^{ab} \left| H - E_0 \right| T + Q \right\rangle \tag{3.89}$$

Further equations coupling triples with singles through pentuples etc.

The total energy is the sum of the reference energy $E_0 = \langle 0 \left| H \right| 0 \rangle$ and the correlation energy

$$E_{\mathrm{C}} = \langle 0 \left| H \right| S + D \rangle \tag{3.90}$$

which requires the exact singles- and doubles amplitudes to be known. In order to truncate the series to singles- and doubles one may either neglect the terms containing the higher excitations on the right hand side (leading to CISD) or approximate their effect thereby losing the variational character of the CI method (CCSD, QCISD and CEPA methods). Defining the one- and two-body excitation operators as $\hat{C}_1 = \sum_{ia} C_a^i a_i^a$, $\hat{C}_2 = \frac{1}{4} \sum_{ijab} C_{ab}^{ij} a_{ij}^{ab}$ one can proceed to approximate the triples and quadruples by the disconnected terms:

$$\left| T \right\rangle = \hat{C}_1 \hat{C}_2 \left| 0 \right\rangle \tag{3.91}$$

$$\left| Q \right\rangle = \frac{1}{2} \hat{C}_2^2 \left| 0 \right\rangle \tag{3.92}$$

As is well known, the CCSD equations contain many more disconnected contributions arising from the various powers of the $\hat{C}_1$ operator (if one would stick to CC logics one would usually label the cluster amplitudes with t_a^i, t_{ab}^{ij},...and the n-body cluster operators with $\hat{T}_n$; we take a CI point of view here). In order to obtain the CEPA type equations from ((3.88)-(3.92)), it is most transparent to relabel the singles and doubles excitations with a compound label P for the internal indices (i) or (*ij*) and x for (a) or (*ab*). Then, the approximations are as follows:

$$\frac{1}{2} \left\langle \Phi_P^x \left| (H - E_0) \hat{C}_2^2 \right| 0 \right\rangle = \frac{1}{2} \sum_{QRyz} C_y^Q C_z^R \left\langle \Phi_P^x \left| H - E_0 \right| \Phi_{QR}^{yz} \right\rangle \tag{3.93}$$

$$\approx C_x^P \sum_{Qy} C_y^Q \left\langle \Phi_P^x \left| H \right| \Phi_{PQ}^{xy} \right\rangle \tag{3.94}$$

$$= C_x^P \sum_{Qy} C_y^Q \left\langle 0 \left| H \right| \Phi_Q^y \right\rangle - C_x^P \sum_{Qy \cup Px} C_y^Q \left\langle 0 \left| H \right| \Phi_Q^y \right\rangle \tag{3.95}$$

$$\approx C_x^P \left(E_{\mathrm{C}} - \sum_{Q \cup P} \varepsilon_Q \right) \tag{3.96}$$

Here the second line contains the approximation that only the terms in which either *Qy* or *Rz* are equal to *Px* are kept (this destroys the unitary invariance) and the fourth line contains the approximation that only "exclusion principle violating" (EPV) terms of internal labels are considered. The notation $Qy \cup Px$ means "*Qy* joint with *Px*" (containing common orbital indices) and ε_Q is the pair correlation energy. The EPV terms must be subtracted from the correlation energy since they arise from double excitations that are impossible due to the fact that an excitation out of an occupied or into an empty orbital of the reference determinant has already been performed. Inserting eq. (3.96) into eq. (3.89) $C_x^P E_C$ cancels and effectively is replaced by the "partial correlation energy" $\sum_{Q \cup P} \varepsilon_Q$.

The resulting equations thus have the appearance of a diagonally shifted ("dressed") CISD equation $\langle \Phi_P^x \left| H - E_0 + \Delta \right| 0 + S + D \rangle = 0$. If the second approximation mentioned above is avoided Malrieu's (SC) ^{2}-CISD arises. [307, 308, 309, 310, 363] Otherwise, one obtains CEPA/3 with the shift:

$$-\Delta_{ab}^{ij} = \sum_k \left(\varepsilon_{ik} + \varepsilon_{jk} \right) - \varepsilon_{ij} \tag{3.97}$$

CEPA/2 is obtained by $-\Delta_{ab}^{ij} = \varepsilon_{ij}$ and CEPA/1 is the average of the CEPA/2 and CEPA/3. As mentioned by Ahlrichs, [364] no consensus appears to exist in the literature for the appropriate shift on the single excitations. If one proceeds straightforwardly in the same way as above, one obtains:

$$\left\langle \Phi_i^a \left| (H - E_0) \hat{C}_1 \hat{C}_2 \right| 0 \right\rangle \approx C_a^i \left(E_{\mathrm{C}} - 2 \sum_k \varepsilon_{ik} \right) \tag{3.98}$$

as the appropriate effect of the disconnected triples on the singles. It has been assumed here that only the singles $\left| \Phi_i^a \right\rangle$ in $\hat{C}_1$ contribute to the shift. If $\left| 0 \right\rangle$ is a HF determinant, the effect of the disconnected triples in the doubles projection vanishes under the same CEPA approximations owing to Brillouin's theorem. Averaged CEPA models are

derived by assuming that all pair correlation energies are equal (except $\varepsilon_{ii} = 0$). As previously discussed by Gdanitz [365], the averaging of CEPA/1 yields $\frac{2}{n}E_{\mathrm{C}}$ and CEPA/3 $E_{\mathrm{C}}\frac{4n-6}{n(n-1)}$ where n is the number of correlated electrons. These happen to be the shifts used for the averaged coupled-pair functional (ACPF [366]) and averaged quadratic coupled-cluster (AQCC [367]) methods respectively. However, averaging the singles shift of eq. (3.98) gives $\frac{4}{n}E_{\mathrm{C}}$. The latter is also the leading term in the expansion of the AQCC shift for large n. In view of the instability of ACPF in certain situations, Gdanitz has proposed to use the AQCC shift for the singles and the original ACPF shift for the doubles and called his new method ACPF/2 [366]. Based on what has been argued above, we feel that it would be most consistent with the ACPF approach to simply use $\frac{4}{n}E_{\mathrm{C}}$ as the appropriate singles shift. We refer to this as NACPF.

It is readily demonstrated that the averaged models may be obtained by a variation of the modified correlation energy functional:

$$E_{\mathrm{C}} = \frac{\langle 0+S+D\,|H-E_0|\,0+S+D\rangle}{1+g_s\,\langle S|S\rangle + g_D\,\langle D|D\rangle} \tag{3.99}$$

with g_S and g_D being the statistical factors $\frac{4}{n}$, $\frac{2}{n}$, $\frac{4n-6}{n(n-1)}$, as appropriate for the given method. Thus, unlike the CEPA models, the averaged models fulfill a stationarity principle and are unitarily invariant. However, if one thinks about localized internal MOs, it appears evident that the approximation of equal pair energies must be one of rather limited validity and that a more detailed treatment of the electron pairs is warranted. Maintaining a stationarity principle while providing a treatment of the pairs that closely resembles that of the CEPA methods was achieved by Ahlrichs and co-workers in an ingenious way with the development of the CPF method [368]. In this method, the correlation energy functional is written as:

$$E_{\mathrm{C}} = 2\sum_{Px}\frac{\left\langle \Phi_p^x\,|H|\,0\right\rangle}{N_P} + \sum_{PQxy}\frac{\left\langle \Phi_p^x\,|H-E_0|\,\Phi_Q^y\right\rangle}{\sqrt{N_P N_Q}} \tag{3.100}$$

with

$$N_P = 1 + \sum_Q T_{PQ}\sum_y (C_y^Q)^2 \tag{3.101}$$

The topological matrix for pairs $P =$(*ij*) and $Q = (kl)$ is chosen as: [115]

$$T_{PQ} = \frac{\delta_{ik}+\delta_{il}}{2n_i} + \frac{\delta_{jk}+\delta_{jl}}{2n_j} \tag{3.102}$$

with n_i being the number of electrons in orbital i in the reference determinant. The singles out of orbital i are formally equated with $P = (ii)$. At the spin-orbital level, $n_i = 1$, for closed shells $n_i = 2$. Using the same topological matrix in $\Delta_P = \sum_Q T_{PQ}\varepsilon_Q$ one recovers the CEPA/1 shifts for the doubles in eq. (3.98). It is straightforward to obtain the CPF equivalents of the other CEPA models by adjusting the T_{PQ} matrix appropriately. In our program, we have done so and we refer below to these methods as CPF/1, CPF/2 and CPF/3 in analogy to the CEPA models (CPF/1 ≡CPF). In fact, as discussed by Ahlrichs and co-workers, variation of the CPF-functional leads to equations that very closely resemble the CEPA equation and can be readily implemented along the same lines as a simple modification of a CISD program. Ahlrichs *et al.* argued that the energies of CEPA/1 and CPF/1 should be very close. We have independently confirmed that in the majority of cases, the total energies predicted by the two methods differ by less than 0.1 mEh.

An alternative to the CPF approach which is also based on variational optimization of an energy functional is the VCEPA method [369]. The equations resulting from application of the variational principle to the VCEPA functional are even closer to the CEPA equations than for CPF so that the resulting energies are practically indistinguishable from the corresponding CEPA values. The VCEPA variants are referred to as VCEPA/1, VCEPA/2, and VCEPA/3 in analogy to CEPA and CPF. A strictly size extensive energy functional (SEOI) which is invariant with respect to unitary transformations within the occupied and virtual orbital subspaces is also available [370] (an open-shell version is not implemented yet).

Again, a somewhat critical point concerns the single excitations. They do not account for a large fraction of the correlation energy. However, large coefficients of the single excitations lead to instability and deterioration of the results. Secondly, linear response properties are highly dependent on the effective energies of the singles and their balanced treatment is therefore important. Since the CEPA and CPF methods amount to shifting down the diagonal energies of the singles and doubles, instabilities are expected if the effective energy of an excitation approaches the

reference energy of even falls below it. In the CPF method this would show up as denominators N_P that are too small. The argument that the CPF denominators are too small has led Chong and Langhoff to the proposal of the MCPF method which uses a slightly more elaborate averaging than $(N_P N_Q)^{1/2}$ [371].[1] However, their modification was solely based on numerical arguments rather than physical or mathematical reasoning. In the light of Eq. (3.98) and the performance of the NACPF, it appears to us that for the singles one should use twice the T_{PQ} proposed by Ahlrichs and co-workers. The topological matrix T_{PQ} is modified in the following way for the (very slightly) modified method to which we refer to as NCPF/1:

$$T_{ij,kl} = \frac{\delta_{ik} + \delta_{il}}{2n_i} + \frac{\delta_{jk} + \delta_{jl}}{2n_j} \tag{3.103}$$

$$T_{ij,k} = 0 \tag{3.104}$$

$$T_{i,kl} = 2\frac{\delta_{ik} + \delta_{il}}{n_i} \tag{3.105}$$

$$T_{i,k} = 0 \tag{3.106}$$

(note that $T_{PQ} \neq T_{QP}$ for this choice). Thus, the effect of the singles on the doubles is set to zero based on the analysis of the CEPA approximations and the effect of the singles on the singles is also set to zero. This is a sensible choice since the product of two single excitations is a double excitation which is already included in the SD space and thus none of them can belong to the outer space. It is straightforward to adapt this reasoning about the single excitations to the CEPA versions as well as to NCPF/2 and NCPF/3.

The aforementioned ambiguities arising from the use of single excitations in coupled-pair methods can be avoided by using correlation-adapted orbitals instead of Hartree-Fock orbitals thus eliminating the single excitations. There are two alternatives: (a) Brueckner orbitals and (b) optimized orbitals obtained from the variational optimization of the electronic energy with respect to the orbitals. Both approaches have already been used for the coupled-cluster doubles (CCD) method [372, 373] and later been extended to coupled-pair methods [374]. In the case of CCD, orbital optimization requires the solution of so-called Λ (or Z vector) equations [375]. There is, however, a cheaper alternative approximating the Z vector by a simple analytical formula [376].

Furthermore, the parametrized coupled-cluster (pCCSD) method of Huntington and Nooijen [377], which combines the accuracy of coupled-pair type methods for (usually superior to CCSD, at least for energies and energy differences) with the higher stability of the coupled-cluster methods, is an attractive alternative. Comprehensive numerical tests [378] indicate that particularly pCCSD(-1,1,1) (or pCCSD/1a) and pCCSD (-1.5,1,1) (or pCCSD/2a) have great potential for accurate computational thermochemistry. These methods can be employed by adding the "simple" keywords `pCCSD/1a` or `pCCSD/2a` to the first line of input. As mentioned in section *Local correlation (DLPNO)*, the DLPNO variants of the pCCSD/1a method is also available for RHF and UHF references via the simple keyword `DLPNO-pCCSD/1a`.

Closed-Shell Equations

Proceeding from spin-orbitals to the spatial orbitals of a closed-shell determinant leads to the actual working equations of this work. Saebo, Meyer and Pulay have exploited the generator state formalism to arrive at a set of highly efficient equations for the CISD problem [379]. A similar set of matrix formulated equations for the CCSD and QCISD cases has been discussed by Werner and co-workers [380] and the MOLPRO implementation is widely recognized to be particularly efficient. Equivalent explicit equations for the CISD and CCSD methods were published by Scuseria *et al.* [381][2] The doubles equations for the residual "vector" σ are ($i \leqslant j$, all a, b):

$$\begin{aligned}
\sigma_{ab}^{ij} = {} & K_{ab}^{ij} + K\left(\mathbf{C}^{ij}\right)_{ab} + \left\{\mathbf{F}^V\mathbf{C}^{ij} + \mathbf{C}^{ij}\mathbf{F}^V\right\}_{ab} - \sum_k \left\{F_{jk}C_{ab}^{ik} + F_{ik}C_{ab}^{kj}\right\} + \sum_{kl} K_{kl}^{ij}C_{ab}^{kl} \\
& + \sum_k \left\{\left(2\mathbf{C}^{ik} - \mathbf{C}^{ik+}\right)\left(\mathbf{K}^{kj} - \tfrac{1}{2}\mathbf{J}^{kj}\right) + \left(\mathbf{K}^{ik} - \tfrac{1}{2}\mathbf{J}^{ik}\right)\left(2\mathbf{C}^{kj} - \mathbf{C}^{kj+}\right)\right\}_{ab} \\
& - \sum_k \left\{\tfrac{1}{2}\mathbf{C}^{ik+}\mathbf{J}^{jk+} + \tfrac{1}{2}\mathbf{J}^{ik}\mathbf{C}^{kj+} + \mathbf{J}^{jk}\mathbf{C}^{ik} + \mathbf{C}^{kj}\mathbf{J}^{ik+}\right\}_{ab} \\
& + C_a^i F_b^j + C_b^j F_a^i - \sum_k \left\{K_{ka}^{ji}C_b^k + K_{kb}^{ij}C_a^k\right\} + \left\{\mathbf{K}^{ia}\mathbf{C}^j + \mathbf{K}^{ja}\mathbf{C}^i\right\}_b \\
& - \Delta^{ij}C_{ab}^{ij}
\end{aligned} \tag{3.107}$$

[1] This method – although it has been rather extensively used in the past – is not implemented in ORCA. We recommend to use our NCPF/1 instead.

[2] Our coupled-cluster implementation is largely based on this nice paper. The equations there have been extensively verified to be correct.

The singles equations are:

$$\begin{aligned}\sigma_a^i = F_a^i + \left\{\mathbf{F}^V \mathbf{C}^i\right\}_a - \sum_j F_{ij} C_a^j - \sum_{jkb} \left(2K_{jb}^{ik} - J_{jb}^{ik}\right) C_{ba}^{kj} \\ + \sum_j \left\{\left(2\mathbf{K}^{ij} - \mathbf{J}^{ij}\right)\mathbf{C}^j + \mathbf{F}^j\left(2\mathbf{C}^{ij+} - \mathbf{C}^{ij}\right) + \left\langle\left(2\mathbf{K}^{ia} - \mathbf{K}^{ia+}\right)\mathbf{C}^{ij+}\right\rangle\right\}_a \\ -\Delta^i C_a^i\end{aligned} \tag{3.108}$$

The following definitions apply:

$$K\left(\mathbf{C}^{ij}\right)_{ab} = \sum_{cd} (ac|bd)\, C_{cd}^{ij} \tag{3.109}$$

$$K_{rs}^{pq} = (pr|qs) \tag{3.110}$$

$$J_{rs}^{pq} = (pq|rs) \tag{3.111}$$

$$\langle \mathbf{AB} \rangle = \sum_{pq} A_{pq} B_{qp} \tag{3.112}$$

The two-electron integrals are written in (11|12) notation and **F** is the closed-shell Fock operator with $\mathbf{F}^V$ being its virtual sub-block. We do not assume the validity of Brillouin's theorem. The amplitudes C_a^i, C_{ab}^{ij} have been collected in vectors $\mathbf{C}^i$ and matrices $\mathbf{C}^{ij}$ wherever appropriate. The shifts Δ^i and Δ^{ij} are dependent on the method used and are defined in Table Table 3.35 for each method implemented in ORCA.

Table 3.35: Summary of the diagonal shifts used in various singles- and doubles methods discussed in this chapter. The quantities ε_i and ε_{ij} are the correlation energy increments brought about by the single- and the double excitations respectively. The partial denominators for the CPF type methods N_i and N_{ij} are specified in eq. (3.101).

Method	Doubles Shift	Singles Shift
CISD	E_C	E_C
CEPA/0	0	0
CEPA/1	$\frac{1}{2}(\varepsilon_i + \varepsilon_j) + \frac{1}{4}\sum_k (\varepsilon_{ik} + \varepsilon_{jk})$	$\frac{1}{2}\varepsilon_{ii} + \frac{1}{2}\sum_k \varepsilon_{ik}$
CEPA/2	$\delta_{ij}\varepsilon_i + \varepsilon_{ij}$	$\varepsilon_i + \varepsilon_{ii}$
CEPA/3	$(\varepsilon_i + \varepsilon_j) - \delta_{ij}\varepsilon_i - \varepsilon_{ij} + \frac{1}{2}\sum_k (\varepsilon_{ik} + \varepsilon_{jk})$	$\varepsilon_i + \sum_k \varepsilon_{ik}$
NCEPA/1	$\frac{1}{4}\sum_k (\varepsilon_{ik} + \varepsilon_{jk})$	$\varepsilon_{ii} + \sum_k \varepsilon_{ik}$
NCEPA/2	ε_{ij}	$2\varepsilon_{ii}$
NCEPA/3	$-\varepsilon_{ij} + \frac{1}{2}\sum_k (\varepsilon_{ik} + \varepsilon_{jk})$	$2\sum_k \varepsilon_{ik}$
CPF/1	$N_{ij}\left\{\frac{1}{2}(\frac{\varepsilon_i}{N_i} + \frac{\varepsilon_j}{N_j}) + \frac{1}{4}\sum_k (\frac{\varepsilon_{ik}}{N_{ik}} + \frac{\varepsilon_{jk}}{N_{jk}})\right\}$	$N_i\left\{\frac{1}{2}\frac{\varepsilon_{ii}}{N_{ii}} + \frac{1}{2}\sum_k \frac{\varepsilon_{ik}}{N_{ik}}\right\}$
CPF/2	$N_{ij}\left\{\delta_{ij}\frac{\varepsilon_i}{N_i} + \frac{\varepsilon_{ij}}{N_{ij}}\right\}$	$N_i\left\{\frac{\varepsilon_i}{N_i} + \frac{\varepsilon_{ii}}{N_{ii}}\right\}$
CPF/3	$N_{ij}\left\{\frac{\varepsilon_i}{N_i}(1-\delta_{ij}) + \frac{\varepsilon_j}{N_j} - \frac{\varepsilon_{ij}}{N_{ij}} + \frac{1}{2}\sum_k (\frac{\varepsilon_{ik}}{N_{ik}} + \frac{\varepsilon_{jk}}{N_{jk}})\right\}$	$N_i\left\{\frac{\varepsilon_i}{N_i} + \sum_k \frac{\varepsilon_{ik}}{N_{ik}}\right\}$
NCPF/1	$\frac{1}{4}N_{ij}\sum_k (\frac{\varepsilon_{ik}}{N_{ik}} + \frac{\varepsilon_{jk}}{N_{jk}})$	$N_i\left\{\frac{\varepsilon_{ii}}{N_{ii}} + \sum_k \frac{\varepsilon_{ik}}{N_{ik}}\right\}$
NCPF/2	$N_{ij}\frac{\varepsilon_{ij}}{N_{ij}}$	$2N_i\frac{\varepsilon_{ii}}{N_{ii}}$
NCPF/3	$N_{ij}\left\{-\frac{\varepsilon_{ij}}{N_{ij}} + \frac{1}{2}\sum_k (\frac{\varepsilon_{ik}}{N_{ik}} + \frac{\varepsilon_{jk}}{N_{jk}})\right\}$	$2N_i\sum_k \frac{\varepsilon_{ik}}{N_{ik}}$
ACPF	$\frac{2}{n}E_\mathrm{C}$	$\frac{2}{n}E_\mathrm{C}$
ACPF/2	$\frac{2}{n}E_\mathrm{C}$	$\left[1 - \frac{(n-3)(n-2)}{n(n-1)}\right]E_\mathrm{C}$
NACPF	$\frac{2}{n}E_\mathrm{C}$	$\frac{4}{n}E_\mathrm{C}$
AQCC	$\left[1 - \frac{(n-3)(n-2)}{n(n-1)}\right]E_\mathrm{C}$	$\left[1 - \frac{(n-3)(n-2)}{n(n-1)}\right]E_\mathrm{C}$

The QCISD method requires some slight modifications. We found it most convenient to think about the effect of the nonlinear terms as a "dressing" of the integrals occurring in equations (3.107) and (3.108). This attitude is close

to the recent arguments of Heully and Malrieu and may even open interesting new routes towards the calculation of excited states and the incorporation of connected triple excitations.[382] The dressed integrals are given by:

$$\bar{F}_{ik} = F_{ik} + \sum_{l} \left\langle \mathbf{C}^{il} \left(2\mathbf{K}^{kl} - \mathbf{K}^{kl+}\right)\right\rangle \tag{3.113}$$

$$\bar{F}_{ab} = F_{ab} - \sum_{kl} \left\{\mathbf{C}^{kl} \left(2\mathbf{K}^{kl} - \mathbf{K}^{kl+}\right)\right\}_{ab} \tag{3.114}$$

$$\bar{F}_{kc} = F_{kc} + \sum_{l} \left(2\mathbf{K}^{kl} - \mathbf{K}^{kl+}\right) \mathbf{C}^{l} \tag{3.115}$$

$$\bar{K}^{ij}_{kl} = K^{ij}_{kl} + \left\langle \mathbf{K}^{kl}\mathbf{T}^{kl+}\right\rangle \tag{3.116}$$

$$\bar{K}^{ij}_{ab} = K^{ij}_{ab} + \sum_{k} \left\{\mathbf{C}^{ik}\left(\mathbf{K}^{kj} - \frac{1}{2}\mathbf{K}^{jk}\right) + \mathbf{C}^{ki}\mathbf{K}^{kj}\right\}_{ab} \tag{3.117}$$

$$\bar{J}^{ij}_{ab} = J^{ij}_{ab} + \sum_{k} \left\{\mathbf{C}^{ki}\mathbf{K}^{jk}\right\}_{ab} \tag{3.118}$$

The CCSD method can be written in a similar way but requires 15 additional terms that we do not document here. They may be taken conveniently from our paper about the LPNO-CCSD method [383].

A somewhat subtle point concerns the definition of the shifts in making the transition from spin-orbitals to spatial orbitals. For example, the CEPA/2 shift becomes in the generator state formalism:

$$-\left\langle \tilde{\Phi}^{ab}_{ij} | \Delta^{ij} | \Psi \right\rangle = C^{ij}_{ab}\left(\frac{1}{3}\varepsilon^{\alpha\alpha}_{ij} + \frac{2}{3}\varepsilon^{\alpha\beta}_{ij}\right) + C^{ij}_{ba}\left(-\frac{1}{3}\varepsilon^{\alpha\alpha}_{ij} + \frac{1}{3}\varepsilon^{\alpha\beta}_{ij}\right) \tag{3.119}$$

($\tilde{\Phi}^{ab}_{ij}$ is a contravariant configuration state function, see Pulay *et al.* [375]. The parallel and antiparallel spin pair energies are given by:

$$\varepsilon^{\alpha\alpha}_{ij} = \frac{1}{2}\sum_{ab}\left[K^{ij}_{ab} - K^{ij}_{ba}\right]\left(C^{ij}_{ab} - C^{ij}_{ba}\right) \tag{3.120}$$

$$\varepsilon^{\alpha\beta}_{ij} = \frac{1}{2}\sum_{ab} K^{ij}_{ab} C^{ij}_{ab} \tag{3.121}$$

This formulation would maintain the exact equivalence of an orbital and a spin-orbital based code. Only in the (unrealistic) case that the parallel and antiparallel pair correlation energies are equal the CEPA/2 shift of Table 3.35 arise. However, we have not found it possible to maintain the same equivalence for the CPF method since the electron pairs defined by the generator state formalism are a combination of parallel and antiparallel spin pairs. In order to maintain the maximum degree of internal consistency we have therefore decided to follow the proposal of Ahlrichs and co-workers and use the topological matrix T_{PQ} in equation (3.102) and the equivalents thereof in the CEPA and CPF methods that we have programmed.

Open-Shell Equations

We have used a non-redundant set of three spin cases ($\alpha\alpha$, $\beta\beta$, $\alpha\beta$) for which the doubles amplitudes are optimized separately. The equations in the spin-unrestricted formalism are straightforwardly obtained from the corresponding spin orbital equations by integrating out the spin. For implementing the unrestricted QCISD and CCSD method, we applied the same strategy (dressed integrals) as in the spin-restricted case. The resulting equations are quite cumbersome and will not be shown here explicitly [384].

Note that the definitions of the spin-unrestricted CEPA shifts differ from those of the spin-restricted formalism described above (see Kollmar et al. [384]). Therefore, except for CEPA/1 and VCEPA/1 (and of course CEPA/0), for which the spin-adaptation of the shift can be done in a consistent way, CEPA calculations of closed-shell molecules yield slightly different energies for the spin-restricted and spin-unrestricted versions. Since variant 1 is also the most accurate among the various CEPA variants [385], we recommend to use variant 1 for coupled-pair type calculations. For the variants 2 and 3, reaction energies of reactions involving closed-shell and open-shell molecules simultaneously should be calculated using the spin-unrestricted versions only.

A subtle point for open-shell correlation methods is the choice of the reference determinant [386]. Single reference correlation methods only yield reliable results if the reference determinant already provides a good description of the systems electronic structure. However, an UHF reference wavefunction suffers from spin-contamination which can spoil the results and lead to convergence problems. This can be avoided if quasi-restricted orbitals (QROs) are used [360, 382] since the corresponding zeroth-order wavefunction is an eigenfunction of the $\hat{S}^2$ operator and thus, no severe spin-contamination will appear. The coupled-pair and coupled-cluster equations will be still solved in a spin-unrestricted formalism but the energy will be slightly higher compared to the results obtained with a spin-polarized UHF reference determinant. Furthermore, especially for more difficult systems like e.g. transition metal complexes, it is often advantageous to use Kohn-Sham (KS) orbitals instead of HF orbitals.

3.10.2 Basic Usage

The coupled-cluster and CI methods in `orca_mdci` are available for RHF and UHF references. The implementation is fairly efficient and suitable for large-scale calculations. The most elementary use of this module is fairly simple.

```
!  METHOD
# where METHOD is:
#   CCSD CCSD(T) QCISD QCISD(T) CPF/n NCPF/n CEPA/n NCEPA/n
#   (n=1,2,3 for all variants) ACPF NACPF AQCC CISD

!  AOX-METHOD
# computes contributions from integrals with 3- and 4-external
# labels directly from AO integrals that are pre-stored in a
# packed format suitable for efficient processing

!  AO-METHOD
# computes contributions from integrals with 3- and 4-external
# labels directly from AO integrals. Can be done for integral
# direct and conventional runs. In particular, the conventional
# calculations can be very efficient

!  MO-METHOD   (this is the default)
# performs a full four index integral transformation. This is
# also often a good choice

!  RI-METHOD
# selects the RI approximation for all integrals. Rarely advisable

!  RI34-METHOD
# selects the RI approximation for the integrals with 3- and 4-
# external labels
#
# The module has many additional options that are documented
# later in the manual.

! RCSinglesFock
! RIJKSinglesFock
! NoRCSinglesFock
! NoRIJKSinglesFock
# Keywords to select the way the so-called singles Fock calculation
# is evaluated. The first two keywords turn on, the second two turn off
# RIJCOSX or RIJK, respectively.
```

Note

- The same FrozenCore options as for MP2 are applied in the MDCI module.
- Since ORCA 4.2, an additional term, called "4th-order doubles-triples correction" is considered in open-shell CCSD(T). To reproduce previous results, one should use a keyword,

```
%mdci
  Include_4thOrder_DT_in_Triples  false
end
```

- The MDCI module cannot deal with systems without alpha virtual orbitals, even if there are beta virtual orbitals. Normally this only happens when the user uses minimal basis sets.

The computational effort for these methods is high — O(N^6) for all methods and O(N^7) if the triples correction is to be computed (calculations based on an unrestricted determinant are roughly 3 times more expensive than closed-shell calculations and approximately six times more expensive if triple excitations are to be calculated). This restricts the calculations somewhat: on presently available PCs 300–400 basis functions are feasible and if you are patient and stretch it to the limit it may be possible to go up to 500–600; if not too many electrons are correlated maybe even up to 800–900 basis functions (when using AO-direct methods).

Tip

- For calculations on small molecules and large basis sets the MO-METHOD option is usually the most efficient; say perhaps up to about 300 basis functions. For integral conventional runs, the AO-METHOD may even more efficient.
- For large calculations (>300 basis functions) the AO-METHOD option is a good choice. If, however, you use very deeply contracted basis sets such as ANOs these calculations should be run in the integral conventional mode.
- AOX-METHOD is usually slightly less efficient than MO-METHOD or AO-METHOD.
- RI-METHOD is seldom the most efficient choice. If the integral transformation time is an issue than you can select `%mdci trafotype trafo_ri` or choose RI-METHOD and then `%mdci kcopt kc_ao`.
- Regarding the singles Fock keywords (RCSinglesFock, etc.), the program usually decides which method to use to evaluate the singles Fock term. For more details on the nature of this term, and options related to its evaluation, see *The singles Fock term*.

To put this into perspective, consider a calculation on serine with the cc-pVDZ basis set — a basis on the lower end of what is suitable for a highly correlated calculation. The time required to solve the equations is listed in Table 3.36. We can draw the following conclusions:

- As long as one can store the integrals and the I/O system of the computer is not the bottleneck, the most efficient way to do coupled-cluster type calculations is usually to go via the full transformation (it scales as O(N^5) whereas the later steps scale as O(N^6) and O(N^7) respectively).
- AO-based coupled-cluster calculations are not much inferior. For larger basis sets (i.e. when the ratio of virtual to occupied orbitals is larger), the computation times will be even more favorable for the AO based implementation. The AO direct method uses much less disk space. However, when you use a very expensive basis set the overhead will be larger than what is observed in this example. Hence, conventionally stored integrals — if affordable — are a good choice.
- AOX-based calculations run at essentially the same speed as AO-based calculations. Since AOX-based calculations take four times as much disk space, they are pretty much outdated and the AOX implementation is only kept for historical reasons.
- RI-based coupled-cluster methods are significantly slower. There are some disk space savings, but the computationally dominant steps are executed less efficiently.
- CCSD is at most 10% more expensive than QCISD. With the latest AO implementation the awkward coupled-cluster terms are handled efficiently.
- CEPA is not much more than 20% faster than CCSD. In many cases CEPA results will be better than CCSD and then it is a real saving compared to CCSD(T), which is the most rigorous.

- If triples are included practically the same comments apply for MO versus AO based implementations as in the case of CCSD.

ORCA is quite efficient in this type of calculation, but it is also clear that the range of application of these rigorous methods is limited as long as one uses canonical MOs. ORCA implements novel variants of the so-called local coupled-cluster method which can calculate large, real-life molecules in a linear scaling time. This will be addressed in Sec. *Local correlation (DLPNO)*.

Table 3.36: Computer times (minutes) for solving the coupled-cluster/coupled-pair equations for Serine (cc-pVDZ basis set)

Method	SCFMode	Time (min)
MO-CCSD	`Conv`	38.2
AO-CCSD	`Conv`	47.5
AO-CCSD	`Direct`	50.8
AOX-CCSD	`Conv`	48.7
RI-CCSD	`Conv`	64.3
AO-QCISD	`Conv`	44.8
AO-CEPA/1	`Conv`	40.5
MO-CCSD(T)	`Conv`	147.0
AO-CCSD(T)	`Conv`	156.7

All of these methods are designed to cover dynamic correlation in systems where the Hartree-Fock determinant dominates the wavefunctions. The least attractive of these methods is CISD which is not size-consistent and therefore practically useless. The most rigorous are CCSD(T) and QCISD(T). The former is perhaps to be preferred, since it is more stable in difficult situations.[3] One can get highly accurate results from such calculations. However, one only gets this accuracy in conjunction with large basis sets. It is perhaps not very meaningful to perform a CCSD(T) calculation with a double-zeta basis set (see Table 3.37). The very least basis set quality required for meaningful results would perhaps be something like def2-TZVP(-f) or preferably def2-TZVPP (cc-pVTZ, ano-pVTZ). For accurate results quadruple-zeta and even larger basis sets are required and at this stage the method is restricted to rather small systems.

Let us look at the case of the potential energy surface of the N_2 molecule. We study it with three different basis sets: TZVP, TZVPP and QZVP. The input is the following:

```
! TZVPP CCSD(T)
%paras R= 1.05,1.13,8
       end
* xyz 0 1
N 0 0 0
N 0 0 {R}
*
```

For even higher accuracy we would need to introduce relativistic effects and - in particular - turn the core correlation on.[4]

Table 3.37: Computed spectroscopic constants of N2 with coupled-cluster methods.

[3] The exponential of the T1 operator serves to essentially fully relax the orbitals of the reference wavefunction. This is not included in the QCISD model that only features at most a linear T1T2 term in the singles residuum. Hence, if the Hartree-Fock wavefunction is a poor starting point but static correlation is not the main problem, CCSD is much preferred over QCISD. This is not uncommon in transition metal complexes.

[4] Note that core correlation is not simply introduced by including the core orbitals in the correlation problem. In addition, special correlation core-polarization functions are needed. They have been standardized for a few elements in the cc-pCVxZ (X=D,T,Q,5,6) basis sets.

Method	Basis set	R_e (pm)	ω_e (cm^{-1})	$\omega_e x_e$ (cm^{-1})
CCSD(T)	SVP	111.2	2397	14.4
	TZVP	110.5	2354	14.9
	TZVPP	110.2	2349	14.1
	QZVP	110.0	2357	14.3
	ano-pVDZ	111.3	2320	14.9
	ano-pVTZ	110.5	2337	14.4
	ano-pVQZ	110.1	2351	14.5
CCSD	QZVP	109.3	2437	13.5
Exp		**109.7**	**2358.57**	**14.32**

One can see from Table 3.37 that for high accuracy - in particular for the vibrational frequency - one needs both - the connected triple-excitations and large basis sets (the TZVP result is fortuitously good). While this is an isolated example, the conclusion holds more generally. If one pushes it, CCSD(T) has an accuracy (for reasonably well-behaved systems) of approximately 0.2 pm in distances, <10 cm^{-1} for harmonic frequencies and a few kcal/mol for atomization energies.[5] It is also astonishing how well the Ahlrichs basis sets do in these calculations — even slightly better than the much more elaborate ANO bases.

Note

The quality of a given calculation is not always high because it carries the label "coupled-cluster". Accurate results are only obtained in conjunction with large basis sets and for systems where the HF approximation is a good 0th order starting point.

Frozen Core Options

In coupled-cluster calculations the Frozen Core (FC) approximation is applied by default. This implies that the core electrons are not included in the correlation treatment, since the inclusion of dynamic correlation in the core electrons usually affects relative energies insignificantly.

The frozen core option can be switched on or off with `! FrozenCore` or `! NoFrozenCore` in the simple input. More information and further options are given in section *Frozen Core Options* and in section *Including (semi)core orbitals in the correlation treatment*.

The singles Fock term

In most MDCI calculations, there is an intermediate, which resembles closely to the SCF Fock matrix, and similar methods are available to efficiently calculate it. In the followings, a short discussion will be given of the so-called singles Fock term, which in the closed shell case has the form

$$G(\mathbf{t}_1)_{pq} = \sum_{jb} t_b^j (2(pq|jb) - (pj|qb)) = \sum_{\mu\nu} c_\mu^p c_\nu^q G(\mathbf{t}_1)_{\mu\nu},$$

The singles Fock matrix can be obtained via transformation from its counterpart ($G(\mathbf{t}_1)_{\mu\nu}$) in the atomic orbital (AO) basis

$$G(\mathbf{t}_1)_{\mu\nu} = \sum_{jb} t_b^j (2(\mu\nu|jb) - (\mu j|\nu b)) = \sum_{\kappa\tau} P(\mathbf{t}_1)_{\kappa\tau} (2(\mu\nu|\kappa\tau) - (\mu\kappa|\nu\tau)), \tag{3.122}$$

where

$$P(\mathbf{t}_1)_{\kappa\tau} = \sum_{jb} t_b^j c_\kappa^j c_\tau^b$$

[5] However, in recent years it became more evident that even CCSD(T) achieves its high apparent accuracy through error cancellations. The full CCSDT method (triples fully included) usually performs worse than CCSD(T). The reason is that the (T) correction undershoots the effects of the triples to some extent and thereby compensates for the neglect of connected quadruple excitations. For very high accuracy quantum chemistry, even these must be considered. The prospects for treating chemically more relevant molecules with such methods is not particularly bright for the foreseeable future…

is the analogue of the SCF density matrix for the singles Fock case. For the singles Coulomb ($J(\mathbf{t}_1)_{\mu\nu}$) case, the density may be symmetrized ($\tilde{P}(\mathbf{t}_1)_{\kappa\tau} = P(\mathbf{t}_1)_{\kappa\tau} + P(\mathbf{t}_1)_{\tau\kappa}$), and one may use the resolution of identity approximation

$$J(\mathbf{t}_1)_{\mu\nu} = \sum_{\kappa\tau} \tilde{P}(\mathbf{t}_1)_{\kappa\tau}(\mu\nu|\kappa\tau) \approx \sum_{AB}\sum_{\kappa\tau} \tilde{P}(\mathbf{t}_1)_{\kappa\tau}(\mu\nu|r_{12}^{-1}|A)V_{AB}^{-1}(B|r_{12}^{-1}|\kappa\tau),$$

where A, B are elements of the RI/DF auxiliary fitting basis. Note that the factor of 2 in ((3.122)) is taken care of by symmetrization. Since we are using a symmetric density, we may use the same efficient routine to evaluate the singles Coulomb term as in the SCF case, see *RI-J* and *Split-RI-J*.

For the exchange case ($K(\mathbf{t}_1)_{\mu\nu}$), one possibility is to use the COSX approximation (see *RIJCOSX*)

$$K(\mathbf{t}_1)_{\mu\nu} = \sum_{\kappa\tau} P(\mathbf{t}_1)_{\kappa\tau}(\mu\kappa|\nu\tau) \approx \sum_g Q_{\mu g} \sum_\tau A_{\nu\tau}(\mathbf{r}_g) \sum_\kappa P(\mathbf{t}_1)_{\kappa\tau} X_{\kappa g},$$

The COSX routine is able to deal with asymmetric densities as well, and thus, it can be used here similar to the SCF case.

The other possibility is to use RI for exchange (RIK),

$$K(\mathbf{t}_1)_{\mu\nu} = \sum_{j\kappa\tau} c_\kappa^j C(\mathbf{t}_1)_\tau^j(\mu\kappa|\nu\tau) \approx \sum_{jAB}(\mu j|r_{12}^{-1}|A)V_{AB}^{-1}(B|r_{12}^{-1}|\nu\tilde{j}),$$

where

$$C(\mathbf{t}_1)_\tau^j = \sum_b t_b^j c_\tau^b,$$

and the $\tilde{j}$ is an "orbital" transformed using $C(\mathbf{t}_1)$.

Using these approximations, there are two approximations for the total singles Fock term, RIJCOSX called by the simple keyword RCSinglesFock and RIJK called by RIJKSinglesFock, see *Basic Usage*. For canonical and LPNO methods, by default the program chooses the same approximation used in the SCF calculation. DLPNO2013 uses RIJCOSX by default, while in DLPNO, the singles Fock term is also evaluated using PNOs via SinglesFockUsePNOs, see *Local correlation (DLPNO)*. This behavior can also be changed by keywords in the method block.

```
%method RIJKSinglesFock 1   # 0 false, 1 true
        RCSinglesFock   0   # 0 false, 1 true
end
```

3.10.3 Coupled-Cluster Densities

If one is mainly accustomed to Hartree-Fock or DFT calculations, the calculation of the density matrix is more or less a triviality and is automatically done together with the solution of the self-consistent field equations. Unfortunately, this is not the case in coupled-cluster theory (and also not in MP2 theory). The underlying reason is that in coupled-cluster theory, the expansion of the exponential $e^{\hat{T}}$ in the expectation value

$$D_{pq} = \frac{\langle\Psi|E_p^q|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{\langle e^{\hat{T}}\Psi_0|E_p^q|e^{\hat{T}}\Psi_0\rangle}{\langle e^{\hat{T}}\Psi_0|e^{\hat{T}}\Psi_0\rangle}$$

only terminates if all possible excitation levels are exhausted, i.e., if all electrons in the reference determinant Ψ_0 (typically the HF determinant) are excited from the space of occupied to the space of virtual orbitals (here D_{pq} denotes the first order density matrix, E_p^q are the spin traced second quantized orbital replacement operators, and $\hat{T}$ is the cluster operator). Hence, the straightforward application of these equations is far too expensive. It is, however, possible to expand the exponentials and only keep the linear term. This then defines a linearized density which coincides with the density that one would calculate from linearized coupled-cluster theory (CEPA/0). The difference to the CEPA/0 density is that converged coupled-cluster amplitudes are used for its evaluation. This density is straightforward to compute and the computational effort for the evaluation is very low. Hence, this is a density that can be easily produced in a coupled-cluster run. It is not, however, what coupled-cluster aficionados would accept as a density.

The subject of a density in coupled-cluster theory is approached from the viewpoint of response theory. Imagine one adds a perturbation of the form

$$H^{(\lambda)} = \lambda \sum_{pq} h^{\lambda}_{pq} E^q_p$$

to the Hamiltonian. Then it is always possible to cast the first derivative of the total energy in the form:

$$\frac{dE}{d\lambda} = \sum_{pq} D^{(\text{response})}_{pq} h^{\lambda}_{pq}$$

This is a nice result. The quantity $D^{(\text{response})}_{pq}$ is the so-called response density. In the case of CC theory where the energy is not obtained by variational optimization of an energy functional, the energy has to be replaced by a Lagrangian reading as follows:

$$L_{CC} = \langle \Phi_0 | \bar{H} | \Phi_0 \rangle + \sum_{\eta} \lambda_{\eta} \langle \Phi_{\eta} | \bar{H} | \Phi_0 \rangle + \sum_{ai} f_{ai} z_{ai}$$

Here $\langle \Phi_{\eta} |$ denotes any excited determinant (singly, doubly, triply, ….). There are two sets of Lagrange multipliers: the quantities z_{ai} that guarantee that the perturbed wavefunction fulfills the Hartree-Fock conditions by making the off-diagonal Fock matrix blocks zero and the quantities λ_{η} that guarantee that the coupled-cluster projection equations for the amplitudes are fulfilled. If both sets of conditions are fulfilled then the coupled-cluster Lagrangian simply evaluates to the coupled-cluster energy. The coupled-cluster Lagrangian can be made stationary with respect to the Lagrangian multipliers z_{ai} and λ_{η}. The response density is then defined through:

$$\frac{dL_{CC}}{d\lambda} = \sum_{pq} D^{(\text{response})}_{pq} \, h^{\lambda}_{pq}$$

The density D_{pq} appearing in this equation does not have the same properties as the density that would arise from an expectation value. For example, the response density can have eigenvalues lower than 0 or larger than 2. In practice, the response density is, however, the best "density" there is for coupled-cluster theory.

Unfortunately, the calculation of the coupled-cluster response density is quite involved because additional sets of equations need to be solved in order to determine the z_{ai} and λ_{η}. If only the equations for λ_{η} are solved one speaks of an "unrelaxed" coupled-cluster density. If both sets of equations are solved, one speaks of a "relaxed" coupled-cluster density. For most intents and purposes, the orbital relaxation effects incorporated into the relaxed density are small for a coupled-cluster density. This is so, because the coupled-cluster equations contain the exponential of the single excitation operator $e^{\hat{T}_1} = \exp(\sum_{ai} t^i_a E^a_i)$. This brings in most of the effects of orbital relaxation. In fact, replacing the $\hat{T}_1$ operator by the operator $\hat{\kappa} = \sum_{ai} \kappa^i_a (E^a_i - E^i_a)$ would provide all of the orbital relaxation thus leading to "orbital optimized coupled-cluster theory" (OOCC).

Not surprisingly, the equations that determine the coefficients λ_{η} (the Lambda equations) are as complicated as the coupled-cluster amplitude equations themselves. Hence, the calculation of the unrelaxed coupled-cluster density matrix is about twice as expensive as the calculation of the coupled-cluster energy (but not quite as with proper program organization terms can be reused and the Lambda equations are linear equations that converge somewhat better than the non-linear amplitude equations).

ORCA features the calculation of the unrelaxed coupled-cluster density on the basis of the Lambda equations for closed- and open-shell systems. If a fully relaxed coupled-cluster density is desired then ORCA still features the orbital-optimized coupled-cluster doubles method (OOCCD). This is not exactly equivalent to the fully relaxed CCSD density matrix because of the operator $\hat{\kappa}$ instead of $\hat{T}_1$. However, results are very close and orbital optimized coupled-cluster doubles is the method of choice if orbital relaxation effects are presumed to be large.

In terms of ORCA keywords, the coupled-cluster density is obtained through the following keywords:

```
#
# coupled-cluster density
#
%mdci  density   none
                 linearized
                 unrelaxed
                 orbopt
       end
```

which will work together with CCSD or QCISD (QCISD and CCSD are identical in the case of OOCCD because of the absence of single excitations). Note, that an unrelaxed density for CCSD(T) is NOT available.

Instead of using the density option "orbopt" in the mdci-block, OOCCD can also be invoked by using the keyword:

```
! OOCCD
```

´

3.10.4 Static versus Dynamic Correlation

Let us look at an "abuse" of the single reference correlation methods by studying (very superficially) a system which is not well described by a single HF determinant. This already occurs for the twisting of the double bond of C_2H_4. At a 90° twist angle the system behaves like a diradical and should be described by a multireference method (see section *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)*)

Fig. 3.4: A rigid scan along the twisting coordinate of C_2H_4. The inset shows the T_1 diagnostic for the CCSD calculation.

As can be seen in Fig. 3.4, there is a steep rise in energy as one approaches a 90° twist angle. The HF curve is actually discontinuous and has a cusp at 90°. This is immediately fixed by a simple CASSCF(2,2) calculation which gives a smooth potential energy surface. Dynamic correlation is treated on top of the CASSCF(2,2) method with the MRACPF approach as follows:

```
#
# twisting the double bond of C2H4
#
! SV(P) def2-TZVP/C SmallPrint NoPop MRACPF
%casscf nel          2
        norb         2
        mult         1
        nroots       1
```

(continues on next page)

(continued from previous page)

```
        TrafoStep  RI
        end
%mrci   tsel 1e-10
        tpre 1e-10
        end
%method scanguess pmodel
        end
%paras  R= 1.3385
        Alpha=0,180,18
        end
* int 0 1
C  0  0  0  0      0   0
C  1  0  0 {R}     0   0
H  1  2  0 1.07  120   0
H  1  2  3 1.07  120 180
H  2  1  3 1.07  120 {Alpha}
H  2  1  3 1.07  120 {Alpha+180}
*
```

This is the reference calculation for this problem. One can see that the RHF curve is far from the MRACPF reference but the CASSCF calculation is very close. Thus, dynamic correlation is not important for this problem! It only appears to be important since the RHF determinant is such a poor choice. The MP2 correlation energy is insufficient in order to repair the RHF result. The CCSD method is better but still falls short of quantitative accuracy. Finally, the CCSD(T) curve is very close the MRACPF. This even holds for the total energy (inset of Fig. 3.5) which does not deviate by more than 2–3 mEh from each other. Thus, in this case one uses the powerful CCSD(T) method in an inappropriate way in order to describe a system that has multireference character. Nevertheless, the success of CCSD(T) shows how stable this method is even in tricky situations. The "alarm" bell for CCSD and CCSD(T) is the so-called "T_1-diagnostic"[6] that is also shown in Fig. 3.5. A rule of thumb says, that for a value of the diagnostic of larger than 0.02 the results are not to be trusted. In this calculation we have not quite reached this critical point although the T_1 diagnostic blows up around the 90° twist.

[6] It is defined as $\|T_1\| / N^{1/2}$ where T_1 are the singles amplitudes and N the number of correlated electrons. The original reference is [415]

Fig. 3.5: Comparison of the CCSD(T) and MRACPF total energies of the C_2H_4 along the twisting coordinate. The inset shows the difference E(MRACPF)-E(CCSD(T)).

The computational cost (disregarding the triples) is such that the CCSD method is the most expensive followed by QCISD (∼10% cheaper) and all other methods (about 50% to a factor of two cheaper than CCSD). The most accurate method is generally CCSD(T). However, this is not so clear if the triples are omitted and in this regime the coupled pair methods (in particular CPF/1 and NCPF/1[7]) can compete with CCSD.

Let us look at the same type of situation from a slightly different perspective and dissociate the single bond of F_2. As is well known, the RHF approximation fails completely for this molecule and predicts it to be unbound. Again we use a much too small basis set for quantitative results but it is enough to illustrate the principle.

We first generate a "reference" PES with the MRACPF method:

```
! def2-SV def2-SVP/C MRACPF
%casscf nel    2
        norb   2
        nroots 1
        mult   1
        end
%mrci   tsel    1e-10
        tpre    1e-10
        end
%paras  R= 3.0,1.3,35
        end
* xyz 0 1
F 0 0 0
F 0 0 {R}
*
```

Note that we scan from outward to inward. This helps the program to find the correct potential energy surface since at large distances the σ and σ^* orbitals are close in energy and fall within the desired 2×2 window for the CASSCF

[7] The "N" methods have been suggested by [416] and are exclusive to ORCA. Please note that our NCPF/1 is different from the MCPF method in the literature [371]. The original CPF method — which we prefer — is from [368]; see also [417] for a nice review about the coupled pair approaches and the underlying philosophy.

calculation (see section *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)*). Comparing the MRACPF and CASSCF curves it becomes evident that the dynamic correlation brought in by the MRACPF procedure is very important and changes the asymptote (loosely speaking the binding energy) by almost a factor of two (see Fig. 3.6). Around the minimum (roughly up to 2.0 Å) the CCSD(T) and MRACPF curves agree beautifully and are almost indistinguishable. Beyond this distance the CCSD(T) calculation begins to diverge and shows an unphysical behavior while the multireference method is able to describe the entire PES up to the dissociation limit. The CCSD curve is qualitatively OK but has pronounced quantitative shortcomings: it predicts a minimum that is much too short and a dissociation energy that is much too high. Thus, already for this rather "simple" molecule, the effect of the connected triple excitations is very important. Given this (rather unpleasant) situation, the behavior of the much simpler CEPA method is rather satisfying since it predicts a minimum and dissociation energy that is much closer to the reference MRACPF result than CCSD or CASSCF. It appears that in this particular case CEPA/1 and CEPA/2 predict the correct result.

Fig. 3.6: Potential energy surface of the F_2 molecule calculated with some single-reference methods and compared to the MRACPF reference.

3.10.5 Basis Sets for Correlated Calculations. The case of ANOs.

In HF and DFT calculations the generation and digestion of the two-electron repulsion integrals is usually the most expensive step of the entire calculation. Therefore, the most efficient approach is to use loosely contracted basis sets with as few primitives as possible — the Ahlrichs basis sets (SVP, TZVP, TZVPP, QZVP, def2-TZVPP, def2-QZVPP) are probably the best in this respect. Alternatively, the polarization-consistent basis sets pc-1 through pc-4 could be used, but they are only available for H-Ar. For large molecules such basis sets also lead to efficient prescreening and consequently efficient calculations.

This situation is different in highly correlated calculations such as CCSD and CCSD(T) where the effort scales steeply with the number of basis functions. In addition, the calculations are usually only feasible for a limited number of basis functions and are often run in the integral conventional mode, since high angular momentum basis functions are present and these are expensive to recompute all the time. Hence, a different strategy concerning the basis set

design seems logical. It would be good to use as few basis functions as possible but make them as accurate as possible. This is compatible with the philosophy of atomic natural orbital (ANO) basis sets. Such basis sets are generated from correlated atomic calculations and replicate the primitives of a given angular momentum for each basis function. Therefore, these basis sets are deeply contracted and expensive but the natural atomic orbitals form a beautiful basis for molecular calculations. In ORCA an accurate and systematic set of ANOs (ano-pVnZ, $n =$ D, T, Q, 5 is incorporated). A related strategy underlies the design of the correlation-consistent basis sets (cc-pVnZ, $n =$ D, T, Q, 5, 6,…) that are also generally contracted except for the outermost primitives of the "principal" orbitals and the polarization functions that are left uncontracted.

Let us study this subject in some detail using the H_2CO molecule at a standard geometry and compute the SCF and correlation energies with various basis sets. In judging the results one should view the total energy in conjunction with the number of basis functions and the total time elapsed. Looking at the data in the Table below, it is obvious that the by far lowest SCF energies for a given cardinal number (2 for double-zeta, 3 for triple zeta and 4 for quadruple-zeta) are provided by the ANO basis sets. Using specially optimized ANO integrals that are available since ORCA 2.7.0, the calculations are not even much more expensive than those with standard basis sets. Obviously, the correlation energies delivered by the ANO bases are also the best of all 12 basis sets tested. Hence, ANO basis sets are a very good choice for highly correlated calculations. The advantages are particularly large for the early members (DZ/TZ).

Table 3.38: Comparison of various basis sets for highly correlated calculations

Basis set	No. Basis Fcns	E(SCF)	E_C(CCSD(T))	E_{tot}(CCSD(T))	Total Time
cc-pVDZ	38	-113.876184	-0.34117952	-114.217364	2
cc-pVTZ	88	-113.911871	-0.42135475	-114.333226	40
cc-pVQZ	170	-113.920926	-0.44760332	-114.368529	695
def2-SVP	38	-113.778427	-0.34056109	-114.118988	2
def2-TZVPP	90	-113.917271	-0.41990287	-114.337174	46
def2-QZVPP	174	-113.922738	-0.44643753	-114.369175	730
pc-1	38	-113.840092	-0.33918253	-114.179274	2
pc-2	88	-113.914256	-0.41321906	-114.327475	43
pc-3	196	-113.922543	-0.44911659	-114.371660	1176
ano-pVDZ	38	-113.910571	-0.35822337	-114.268795	12
ano-pVTZ	88	-113.920389	-0.42772994	-114.348119	113
ano-pVQZ	170	-113.922788	-0.44995355	-114.372742	960

Fig. 3.7: Error in Eh for various basis sets for highly correlated calculations relative to the ano-pVQZ basis set.

Let us look at one more example in Table 3.39: the optimized structure of the N_2 molecule as a function of basis set using the MP2 method *(these calculations are a bit older from the time when the ano-pVnZ basis sets did not yet exist. Today, the ano-pVnZ would be preferred)* .

The highest quality basis set here is QZVP and it also gives the lowest total energy. However, this basis set contains up to g-functions and is very expensive. Not using g-functions and a set of f-functions (as in TZVPP) has a noticeable effect on the outcome of the calculations and leads to an overestimation of the bond distance of 0.2 pm — a small change but for benchmark calculations of this kind still significant. The error made by the TZVP basis set that lacks the second set of d-functions on the bond distance, binding energy and ionization potential is surprisingly small even though the deletion of the second d-set "costs" more than 20 mEh in the total energy as compared to TZV(2d,2p), and even more compared to the larger TZVPP.

A significant error on the order of 1 – 2 pm in the calculated distances is produced by smaller DZP type basis sets, which underlines once more that such basis sets are really too small for correlated molecular calculations — the ANO-DZP basis sets are too strongly biased towards the atom, while the "usual" molecule targeted DZP basis sets like SVP have the d-set designed to cover polarization but not correlation (the correlating d-functions are steeper than the polarizing ones). The performance of the very economical SVP basis set should be considered as very good, and (a bit surprisingly) slightly better than cc-pVDZ despite that it gives a higher absolute energy.

Essentially the same picture is obtained by looking at the (uncorrected for ZPE) binding energy calculated at the MP2 level – the largest basis set, QZVP, gives the largest binding energy while the smaller basis sets underestimate it. The error of the DZP type basis sets is fairly large ($\approx$ 2 eV) and therefore caution is advisable when using such bases.

Table 3.39: Comparison of various basis sets for correlated calculations.

Basis set	R_{eq} (pm)	E(2N-N_2) (eV)	IP(N/N^+) (eV)	E(MP2) (Eh)
SVP	112.2	9.67	14.45	**-109.1677**
cc-pVDZ	112.9	9.35	14.35	**-109.2672**
TZVP	111.5	10.41	14.37	**-109.3423**
TZV(2d,2p)	111.4	10.61	14.49	**-109.3683**
TZVPP	111.1	10.94	14.56	**-109.3973**
QZVP	110.9	11.52	14.60	**-109.4389**

3.10.6 The Coupled Cluster S-Diagnostic

There have been a number of diagnostic proposed for coupled cluster calculations that are designed to indicate "multi-reference" character. Unfortunately, the most widely used of them is the "T1-diagnostic". We have argued for a long time, that the T1-diagnostic has nothing to do with multi-reference character. Rather, it measures the extent of orbital relaxation in the dynamic correlation field. While this is a measure of how good or bad the Hartree-Fock approximation works for the given system, it is expressis verbis NOT related to the multi-configuration character that the CC wavefunction may or may not want to emulate. For a discussion of what we consider as multi-reference versus single-reference character, please consult the paper by Izsak et al.

ORCA has an alternative coupled cluster diagnostic, known as the "S-diagnostic" that was introduced by Perdersen et al. This diagnostic does indeed measure multi-reference character. It is available for canonical RHF and UHF CCSD and is invoked by

```
%mdci DoSDiagonstic true
end
```

Relevant Papers:

1. Liakos, D. G.; Neese, F. Interplay of Correlation and Relativistic Effects in Correlated Calculations on Transition-Metal Complexes: The $Cu_2O_2^{2+}$ Core Revisited. *J. Chem. Theory Comput.*, **2011**, 7, 1511–1523.

2. Izsák, Róbert; Ivanov, Aleksei V.; Blunt, Nick S.; Holzmann, Nicole; Neese, Frank. Measuring Electron Correlation: The Impact of Symmetry and Orbital Transformations. *Journal of Chemical Theory and Computation*, **2023**, 19 (10), 2703–2720. PMID: 37022051. arXiv:https://doi.org/10.1021/acs.jctc.3c00122, DOI: 10.1021/acs.jctc.3c00122.

3. Faulstich, Fabian M.; Kristiansen, Håkon E.; Csirik, Mihaly A.; Kvaal, Simen; Pedersen, Thomas Bondo; Laestadius, Andre. S-Diagnostic—An a Posteriori Error Assessment for Single-Reference Coupled-Cluster Methods. *The Journal of Physical Chemistry A*, **2023**, 127 (43), 9106–9120. PMID: 37874274. arXiv:https://doi.org/10.1021/acs.jpca.3c01575, DOI: 10.1021/acs.jpca.3c01575.

3.10.7 Explicitly Correlated Methods: F12-MP2 and F12-CCSD(T)

A physically perhaps somewhat more satisfying alternative to basis set extrapolation is the theory of explicit correlation. In this method terms are added to the wavefunction Ansatz that contain the interelectronic coordinates explicitly (hence the name "explicit correlation"). Initially these terms were linear in the interelectronic distances ("R12-methods"). However, it has later been found that better results can be obtained by using other functions, such as an exponential, of the interelectronic distance ("F12-methods"). These methods are known to yield near basis set limit results for correlation energies in conjunction with much smaller orbital basis sets.

In applying these methods several points are important:

- Special orbital basis sets are at least advantageous. The development of such basis sets is still in its infancy. For a restricted range of elements the basis sets cc-pVnZ-F12 are available (where n = D, T, Q) and are recommended. Note, that other than their names suggest, these are a fair bit larger than regular double, triple or quadruple-zeta basis sets
- In addition to an orbital basis set, a near-complete auxiliary basis set must be specified. This is the so-called "CABS" basis. For the three basis sets mentioned above these are called cc-pVnZ-F12-CABS. If you have elements that are not covered you are on your own to supply a CABS basis set. CABS basis sets can be read into ORCA in a way analogous to RI auxiliary basis sets (replace "AUX" by "CABS" in the input). There are automatic tools for building a CABS basis from an arbitrary orbital basis, e.g. AutoCABS[387]
- if the RI approximation is used in conjunction with F12, a third basis set is required - this can be the regular auxiliary "/C" basis, but we recommend to step one level up in the auxiliary basis set (e.g. use a cc-pVTZ/C fitting basis in conjunction with cc-pVDZ-F12)
- It is perfectly feasible to use RIJCOSX or RI-JK at the same time. In this case, you should provide a fourth basis set for the Coulomb fitting
- RHF and UHF are available, ROHF not. (Although, one can do a ROHF like calculation by using QROs)
- Gradients are not available

Doing explicitly correlated MP2 calculations is straightforward. For example look at the following calculation on the water molecule at a given geometry:

```
#
! F12-MP2 cc-pVDZ-F12 cc-pVDZ-F12-CABS  VeryTightSCF PModel

* xyz 0 1
    O    0.000000000000     0.000000000000     0.369372944000
    H    0.783975899000     0.000000000000    -0.184686472000
    H   -0.783975899000     0.000000000000    -0.184686472000
*
```

and similarly in conjunction with the RI approximation:

```
#
! F12-RI-MP2 cc-pVDZ-F12 cc-pVDZ-F12-CABS cc-pVTZ/C VeryTightSCF PModel

* xyz 0 1
    O    0.000000000000     0.000000000000     0.369372944000
    H    0.783975899000     0.000000000000    -0.184686472000
    H   -0.783975899000     0.000000000000    -0.184686472000
*
```

The output is relatively easy to interpret:

```
-----------------
RI-MP2-F12 ENERGY
-----------------

EMP2 correlation Energy            :      -0.241038994909
F12 correction                     :      -0.054735459470
```

(continues on next page)

(continued from previous page)

```
                                          -----------------
MP2 basis set limit estimate       :      -0.295774454379

Hartree-Fock energy                :     -76.057963800414
(2)_S CABS correction to EHF       :      -0.003475342535
                                          -----------------
HF basis set limit estimate        :     -76.061439142949

MP2 total energy before F12        :     -76.299002795323
Total F12 correction               :      -0.058210802005
                                          -----------------
Final basis set limit MP2 estimate :     -76.357213597328
```

It consists of several parts. The first is the regular (RI-)MP2 correlation energy in the orbitals basis followed by the additive MP2 correction which are combined to provide an MP2 correlation energy basis set limit estimate. The second part consists of an estimate in the error in the underlying SCF energy. This is the "(2)_S CABS" correction. The combination of the SCF energy with this correction yields an estimate of the SCF basis set limit. The correction will typically undershoot somewhat, but the error is very smooth. Finally, the corrected correlation energy and the corrected SCF energy are added to yield the F12 total energy estimate at the basis set limit.

Let's look at some results and compare to extrapolation:

```
#
# Correlation energies of the water molecule: extrapolation versus F12
#
# cc-pVDZ MP2: -0.201380894
#      T     : -0.261263141
#      Q     : -0.282661311
#     T/Q    : -0.298276192
#     Q/5    : -0.300598282
#  F12-DZ    : -0.295775804
#  RI-F12-DZ : -0.295933560 (cc-pVDZ/C)
#              -0.295774489 (cc-pVTZ/C)
#  F12-TZ    : -0.299164006
#  RI-F12-TZ : -0.299163478 (cc-pVQZ/C)
#  F12-QZ    : -0.300130086
```

It is obvious that extrapolated and F12 correlation energies converge to the same number (in this case around 300 mEh). The best extrapolated result is still below the F12 result (this would primarily be meaningful in a variational calculation). However, first of all this was an expensive extrapolation and second, the small residual F12 error is very smooth and cancels in energy differences. In any case, already the F12-double-zeta (where "double zeta" is to be interpreted rather loosely) brings one into within 5 mEh of the basis set limit correlation energy and the F12-triple-zeta calculation to within 1 mEh, which is impressive.

The additional effort for the F12 calculation is rather high, since five types of additional two-electron integrals need to be calculated. Both integrals in CABS space and in the original orbital (OBS) space must be calculated and mixed Fock matrices are also required. Hence, one may wonder, whether a double-zeta F12 calculation actually saves any time over, say, a quadruple-zeta regular calculation. The actual answer to this question is: "NO". Given all possibilities of obtained approximate MP2 and SCF energies, we have investigated the question of how to obtain MP2 basis set limit energies most efficiently in some detail. The results show that in terms of timings, basis set extrapolation in combination with RI-JK is the method of choice for MP2.[388] However, energy differences are more reliable with F12-MP2. In combination with RI-JK or RIJCOSX F12-MP2 becomes also competitive in terms of computational efficiency.

This situation is different in the case of coupled-cluster methods, where F12 methods outperform extrapolation and are the method of choice.

For coupled-cluster theory, everything works in a very similar fashion:

```
# the keywords
! F12-CCSD(T)
```

(continues on next page)

(continued from previous page)

```
# and
! CCSD(T)-F12
# are equivalent
```

A special feature of ORCA that can save large amounts of time, is to use the RI approximation only for the F12-part. The keyword here is:

```
! F12/RI-CCSD(T)
# or
! CCSD(T)-F12/RI
```

Everything else works as described for F12-MP2.

3.10.8 Automatic extrapolation to the basis set limit

Note

- This functionality is deprecated - it may still be usable but we will not actively maintain this part of code anymore. For basis set extrapolation please use the respective compound scripts.

As eluded to in the previous section, one of the biggest problems with correlation calculations is the slow convergence to the basis set limit. One possibility to overcome this problem is the use of explicitly correlated methods. The other possibility is to use basis set extrapolation techniques. Since this involves some fairly repetitive work, some procedures were hardwired into the ORCA program. So far, only energies are supported. For extrapolation, a systematic series of basis sets is required. This is, for example, provided by the cc-pVnZ, aug-cc-pVnZ or the corresponding ANO basis sets. Here n is the "cardinal number" that is 2 for the double-zeta basis sets, 3 for triple-zeta, etc.

The convergence of the HF energy to the basis set limit is assumed to be given by:

$$E_{\text{SCF}}^{(X)} = E_{\text{SCF}}^{(\infty)} + A \exp\left(-\alpha\sqrt{X}\right) \tag{3.123}$$

Here, $E_{\text{SCF}}^{(X)}$ is the SCF energy calculated with the basis set with cardinal number X, $E_{\text{SCF}}^{(\infty)}$ is the basis set limit SCF energy and A and α are constants. The approach taken in ORCA is to do a two-point extrapolation. This means that either A or α have to be known. Here, we take A as to be determined and α as a basis set specific constant.

The correlation energy is supposed to converge as:

$$E_{\text{corr}}^{(\infty)} = \frac{X^\beta E_{\text{corr}}^{(X)} - Y^\beta E_{\text{corr}}^{(Y)}}{X^\beta - Y^\beta} \tag{3.124}$$

The theoretical value for β is 3.0. However, it was found by Truhlar and confirmed by us, that for 2/3 extrapolations $\beta = 2.4$ performs considerably better.

For a number of basis sets, we have determined the optimum values for α and β[48]:

	α_{23}	β_{23}	α_{34}	β_{34}
cc-pV*n*Z	4.42	2.46	5.46	3.05
pc-*n*	7.02	2.01	9.78	4.09
def2	10.39	2.40	7.88	2.97
ano-pV*n*Z	5.41	2.43	4.48	2.97
saug-ano-pV*n*Z	5.48	2.21	4.18	2.83
aug-ano-pV*n*Z	5.12	2.41		

Since the β values for 2/3 are close to 2.4, we always take this value. Likewise, all 3/4 and higher extrapolations are done with $\beta = 3$. However, the optimized values for α are taken throughout.

Using the keyword `! Extrapolate(X/Y,basis)`, where `X` and `Y` are the corresponding successive cardinal numbers and `basis` is the type of basis set requested (= `cc`, `aug-cc`, `cc-core`, `ano`, `saug-ano`, `aug-ano`, `def2`) ORCA will calculate the SCF and optionally the MP2 or MDCI energies with two basis sets and separately extrapolate.

The keyword works also in the following way: `! Extrapolate(n,basis)` where `n` is the is the number of energies to be used. In this way the program will start from a double-zeta basis and perform calculations with `n` cardinal numbers and then extrapolate the different pairs of basis sets. Thus for example the keyword `! Extrapolate(3, CC)` will perform calculations with cc-pVDZ, cc-pVTZ and cc-pVQZ basis sets and then estimate the extrapolation results of both cc-pVDZ/cc-pVTZ and cc-pVTZ/cc-pVQZ combinations.

Let us take the example of the H2O molecule at the B3LYP/TZVP optimized geometry. The reference values have been determined from a HF calculation with the decontracted aug-cc-pV6Z basis set and the correlation energy was obtained from the cc-pV5Z/cc-pV6Z extrapolation. This gives:

```
E(SCF,CBS)            = -76.066958 Eh
EC(CCSD(T),CBS)       =  -0.30866 Eh
Etot(CCSD(T),CBS)     = -76.37561 Eh
```

Now we can see what extrapolation can bring in:

```
!CCSD(T) Extrapolate(2/3) TightSCF Conv Bohrs
* int 0 1
O 0 0 0  0 0 0
H 1 0 0  1.81975 0 0
H 1 2 0  1.81975 105.237 0
*
```

NOTE:

- The RI-JK and RIJCOSX approximations work well together with this option and RI-MP2 is also possible. Auxiliary basis sets are automatically chosen and can not be changed.
- All other basis set choices, externally defined bases etc. will be ignored — the automatic procedure only works with the default basis sets!
- The basis sets with the "core" postfix contain core correlation functions. By default it is assumed that this means that the core electrons are also to be correlated and the frozen core approximation is turned off. However, this can be overridden in the method block by choosing, e.g. `%method frozencore fc_electrons end`!
- So far, the extrapolation is only implemented for single points and not for gradients. Hence, geometry optimizations cannot be done in this way.
- The extrapolation method should only be used with very tight SCF convergence criteria. For open shell methods, additional caution is advised.

This gives:

```
Alpha(2/3)      :  4.420 (SCF Extrapolation)
Beta(2/3)       :  2.460 (correlation extrapolation)

SCF energy with basis cc-pVDZ:                          -76.026430944
SCF energy with basis cc-pVTZ:                          -76.056728252
Extrapolated CBS SCF energy (2/3) :                     -76.066581429 (-0.
↪009853177)

MDCI energy with basis cc-pVDZ:                          -0.214591061
MDCI energy with basis cc-pVTZ:                          -0.275383015
Extrapolated CBS correlation energy (2/3) :              -0.310905962 (-0.
↪035522947)
```

(continues on next page)

(continued from previous page)

```
Estimated CBS total energy (2/3) :                        -76.377487391
```

Thus, the error in the total energy is indeed strongly reduced. Let us look at the more rigorous 3/4 extrapolation:

```
Alpha(3/4)      :  5.460 (SCF Extrapolation)
Beta(3/4)       :  3.050 (correlation extrapolation)

SCF energy with basis cc-pVTZ:                            -76.056728252
SCF energy with basis cc-pVQZ:                            -76.064381269
Extrapolated CBS SCF energy (3/4) :                       -76.066687152 (-0.
↪002305884)

MDCI energy with basis cc-pVTZ:                            -0.275383016
MDCI energy with basis cc-pVQZ:                            -0.295324345
Extrapolated CBS correlation energy (3/4) :                -0.309520368 (-0.
↪014196023)

Estimated CBS total energy (3/4) :                        -76.376207520
```

In our experience, the ANO basis sets extrapolate similarly to the cc-basis sets. Hence, repeating the entire calculation with `Extrapolate(3,ANO)` gives:

```
Estimated CBS total energy (2/3) :                        -76.377652792
Estimated CBS total energy (3/4) :                        -76.376983433
```

Which is within 1 mEh of the estimated CCSD(T) basis set limit energy in the case of the 3/4 extrapolation and within 2 mEh for the 2/3 extrapolation.

For larger molecules, the bottleneck of the calculation will be the CCSD(T) calculation with the larger basis set. In order to avoid this expensive (or prohibitive) calculation, it is possible to estimate the CCSD(T) energy at the basis set limit as:

$$E_{\text{corr}}^{(\text{CCSD(T)};Y)} \approx E_{\text{corr}}^{(\text{CCSD(T)};X)} + E_{\text{corr}}^{(\text{MP2};\infty)} - E_{\text{corr}}^{(\text{MP2};X)} \tag{3.125}$$

This assumes that the basis set dependence of MP2 and CCSD(T) is similar. One can then extrapolate as before. Alternatively, the standard way — as extensively exercised by Hobza and co-workers — is to simply use:

$$E_{\text{total}}^{(\text{CCSD(T)};\text{CBS})} \approx E_{\text{SCF}}^{(Y)} + E_{\text{corr}}^{(\text{CCSD(T)};X)} + E_{\text{corr}}^{(\text{MP2};\infty)} - E_{\text{corr}}^{(\text{MP2};X)} \tag{3.126}$$

The appropriate keyword is:

```
! ExtrapolateEP2(2/3,ANO,MP2) TightSCF Conv Bohrs
* int 0 1
O 0 0 0  0 0 0
H 1 0 0  1.81975 0 0
H 1 2 0  1.81975 105.237 0
*
```

This creates the following output:

```
Alpha      :  5.410 (SCF Extrapolation)
Beta       :  2.430 (correlation extrapolation)

SCF energy with basis ano-pVDZ:                           -76.059178452
SCF energy with basis ano-pVTZ:                           -76.064774379
Extrapolated CBS SCF energy :                             -76.065995735 (-0.
↪001221356)

MP2 energy with basis ano-pVDZ:                            -0.219202871
MP2 energy with basis ano-pVTZ:                            -0.267058634
```

(continues on next page)

(continued from previous page)

```
Extrapolated CBS correlation energy :                    -0.295568604 (-0.
↪028509970)

CCSD(T) correlation energy with basis ano-pVDZ:          -0.229478341
CCSD(T) - MP2 energy with basis ano-pVDZ:                -0.010275470

Estimated CBS total energy :                            -76.371839809
```

The estimated correlation energy is not really bad — within 3 mEh from the basis set limit.

Using the `ExtrapolateEP2(n/m,bas,[method, method-details])` keyword one can use a generalization of the above method where instead of MP2 any available correlation method can be used as described in Ref. [389]. `method` is optional and can be either MP2 or DLPNO-CCSD(T), the latter being the default. In case the method is DLPNO-CCSD(T) in the `method-details` option one can ask for LoosePNO, NormalPNO or TightPNO.

$$E_{\text{corr}}^{(\text{CCSD(T)};CBS)} \approx E_{\text{corr}}^{(\text{CCSD(T)};X)} + E_{\text{corr}}^{(\text{M};CBS)}(X, X+1) - E_{\text{corr}}^{(\text{M};X)} \tag{3.127}$$

Here M represents any correlation method one would like to use. For the previous water molecule the input of a calculation that uses DLPNO-CCSD(T) (which is the default now) instead of MP2 would look like:

```
! ExtrapolateEP2(2/3,cc,DLPNO-CCSD(T)) TightSCF Conv Bohrs
* int 0 1
O 0 0 0  0 0 0
H 1 0 0  1.81975 0 0
H 1 2 0  1.81975 105.237 0
*
```

and it would produce the following output:

```
Alpha     :  4.420 (SCF Extrapolation)
Beta      :  2.460 (correlation extrapolation)

SCF energy with basis cc-pVDZ:                          -76.026430944
SCF energy with basis cc-pVTZ:                          -76.056728252
Extrapolated CBS SCF energy :                           -76.066581429 (-0.
↪009853177)

MDCI energy with basis cc-pVDZ:                          -0.214429497
MDCI energy with basis cc-pVTZ:                          -0.275299699
Extrapolated CBS correlation energy :                    -0.310868368 (-0.
↪035568670)

CCSD(T) correlation energy with basis cc-pVDZ:           -0.214548320
CCSD(T) - MDCI energy with basis cc-pVDZ:                -0.000118824

Estimated CBS total energy :                            -76.377568621
```

which is less than 2 mEh from the basis set limit. Finally it was shown [389] that instead of extrapolating the cheap method, M, using cardinal numbers X and $X+1$ it is better to use cardinal numbers $X+1$ and $X+2$.

$$E_{\text{corr}}^{(\text{CCSD(T)};CBS)} \approx E_{\text{corr}}^{(\text{CCSD(T)};X)} + E_{\text{corr}}^{(\text{M};CBS)}(X+1, X+2) - E_{\text{corr}}^{(\text{M};X)} \tag{3.128}$$

This can be done using the `ExtrapolateEP3(bas,[method,method-details])` keyword:

```
! ExtrapolateEP3(CC) TightSCF Conv Bohrs
```

and the corresponding output would be:

```
Alpha     :  5.460 (SCF Extrapolation)
Beta      :  3.050 (correlation extrapolation)
```

(continues on next page)

(continued from previous page)

```
SCF energy with basis cc-pVDZ:                          -76.026430944
SCF energy with basis cc-pVTZ:                          -76.056728252
SCF energy with basis cc-pVQZ:                          -76.064381269
Extrapolated CBS SCF energy :                           -76.066687152 (-0.
↪002305884)

MDCI energy with basis cc-pVDZ:                          -0.214429497
MDCI energy with basis cc-pVTZ:                          -0.275299699
MDCI energy with basis cc-pVQZ:                          -0.295229871
Extrapolated CBS correlation energy :                    -0.309417951 (-0.
↪014188080)

CCSD(T) correlation energy with basis cc-pVDZ:           -0.214548319
CCSD(T) - MDCI energy with basis cc-pVDZ:                -0.000118822

Estimated CBS total energy :                            -76.376223926
```

For the ExtrapolateEP2, and ExtrapolateEP3 keywords the default cheap method is the DLPNO-CCSD(T) with the NormalPNO thresholds. There also available options with MP2, and DLPNO-CCSD(T) with LoosePNO and TightPNO settings.

3.10.9 Cluster in molecules (CIM)

Cluster in molecules (CIM) approach is a linear scaling local correlation method developed by Li and the coworkers in 2002.[390] It was further improved by Li, Piecuch, Kállay and other groups recently.[391, 392, 393, 394, 395] The CIM is inspired by the early local correlation method developed by Förner and coworkers.[396] The total correlation energy of a closed-shell molecule can be considered as a summation of correlation energies of each occupied LMOs.

$$E_{\text{corr}} = \sum_{i}^{occ} E_i = \sum_{i}^{occ} \frac{1}{4} \sum_{j,ab} \langle ij||ab\rangle T_{ab}^{ij} \tag{3.129}$$

For each occupied LMO, it only correlates with its nearby occupied LMOs and virtual MOs. To reproduce the correlation energy of each occupied LMO, only a subset of occupied and virtual LMOs are needed in the correlation calculation. Instead of doing the correlation calculation of the whole molecule, the correlation energies of all LMOs can be obtained within various subsystems.

The CIM approach implemented in ORCA is following an algorithm proposed by Guo and coworkers with a few improvements.[394, 395]

1. To avoid the real space cutoff, the differential overlap integral (DOI) is used instead of distance threshold. There is only one parameter 'CIMTHRESH' in CIM approach, controlling the construction of CIM subsystems. If the DOI between LMO *i* and LMO *j* is larger than CIMTHRESH, LMO *j* will be included into the MO domain of *i*. By including all nearby LMO of *i*, one can construct a subsystem for MO *i*. The default value of CIMTHRESH is 0.001. If accurate results are needed, a tighter CIMTHRESH must be used.
2. Since ORCA 4.1, the neglected correlations between LMO *i* and LMOs outside the MO domain of *i* are considered as well. These weak correlations are approximately evaluated by dipole moment integrals. With this correction, the CIM results of 3 dimensional proteins are significantly improved. About 99.8% of the correlation energies are recovered.

The CIM can invoke different single reference correlation methods for the subsystem calculations. In ORCA the CIM-RI-MP2, CIM-CCSD(T), CIM-DLPNO-MP2 and CIM-DLPNO-CCSD(T) methods are available. The CIM-RI-MP2 and CIM-DLPNO-CCSD(T) have been proved to be very efficient and accurate methods to compute correlation energies of very big molecules, containing a few thousand atoms.[395]
The usage of CIM in ORCA is simple. For CIM-RI-MP2,

```
#
# CIM-RI-MP2 calculation
```

(continues on next page)

(continued from previous page)

```
#
! RI-MP2 cc-pVDZ cc-pVDZ/C CIM
%CIM
  CIMTHRESH 0.0005  # Default value is 0.001
end

* xyzfile 0 1 CIM.xyz
```

For CIM-DLPNO-CCSD(T),

```
#
# CIM-DLPNO-CCSD calculation
#
! DLPNO-CCSD(T) cc-pVDZ  cc-pVDZ/C CIM
* xyzfile 0 1 CIM.xyz
```

The parallel efficiency of CIM has been significantly improved.[395] Except for a few domain construction sub-steps, the CIM algorithm can achieve very high parallel efficiency. Since ORCA 4.1, the parallel version does not support Windows platform anymore due to the parallelization strategy. The generalization of CIM from closed-shell to open-shell (multi-reference) will also be implemented in the near future.

3.10.10 Local correlation (DLPNO)

ORCA features the extremely powerful "domain-based local pair natural orbital" approximation (DLPNO),[397] which us an extension of the "local pair natural orbital" ansatz (LPNO).[383, 384, 385, 398] These methods are designed to provide results as close as possible to the canonical coupled-cluster results while gaining orders of magnitude of efficiency through a series of well-controlled approximations. In fact, typically about 99.8% to 99.9% of the canonical correlation energy is recovered in such calculations. Even higher accuracy is achievable but will ultimately come at much higher computational cost. The default cut-offs are set such that the vast majority of chemically meaningful energy differences are reproduced to better than 1 kcal/mol relative to the canonical results with the same basis set.

With ORCA 6.0, the LPNO variants are no longer supported, as they involve some higher order scaling steps (up to N^5) while DLPNO is linear scaling and of similar accuracy. The comparison between LPNO-CCSD and DLPNO-CCSD is shown in Fig. 3.8. It is obvious that DLPNO-CCSD is (almost) never slower than LPNO-CCSD. However, its true advantages do become most apparent for molecules with more than approximately 60 atoms. The triples correction, that was added with our second paper from 2013, shows a perfect linear scaling, as is shown in part (a) of Fig. 3.8. For large systems it adds about 10%–20% to the DLPNO-CCSD computation time, hence its addition is possible for all systems for which the latter can still be obtained. Since 2016, the entire DLPNO-CCSD(T) algorithm is linear scaling. The improvements of the linear-scaling algorithm, compared to DLPNO2013-CCSD(T), start to become significant at system sizes of about 300 atoms, as becomes evident in part (b) of Fig. 3.8.

(a) (a) DLPNO2013 Scaling

(b) (b) DLPNO Scaling

Fig. 3.8: a) Scaling behavior of the canonical CCSD, LPNO-CCSD and DLPNO2013-CCSD(T) methods. It is obvious that only DLPNO2013-CCSD and DLPNO2013-CCSD(T) can be applied to large molecules. The advantages of DLPNO2013-CCSD over LPNO-CCSD do not show before the system has reached a size of about 60 atoms. b) Scaling behavior of DLPNO2013-CCSD(T), DLPNO-CCSD(T) and RHF using RIJCOSX. It is obvious that only DLPNO-CCSD(T) can be applied to truly large molecules, is faster than the DLPNO2013 version, and even has a crossover with RHF at about 400 atoms.

Using the DLPNO-CCSD(T) approach it was possible for the first time (in 2013) to perform a CCSD(T) level calculation on an entire protein (Crambin with more than 650 atoms, Fig. 3.9). While the calculation using a double-zeta basis took about 4 weeks on one CPU with DLPNO2013-CCSD(T), it takes only about 4 days to complete with DLPNO-CCSD(T). With DLPNO-CCSD(T) even the triple-zeta basis calculation can be completed within reasonable time, taking 2 weeks on 4 CPUs.

Fig. 3.9: Structure of the Crambin protein - the first protein to be treated with a CCSD(T) level ab initio method

It is important to understand that the LPNO and DLPNO implementations are intimately tied to the RI approximation. Hence, in these calculations one *must* specify a fitting basis set. The same rules as for RI-MP2 apply: the auxiliary basis sets optimized for MP2 are just fine for PNO calculations. You can specify several aux bases for the same job and the program will sort it out correctly.

The theory of the LPNO and DLPNO methods has been thoroughly described in the literature in a number of original research papers:

- F. Neese, A. Hansen, D. G. Liakos: Efficient and accurate local approximations to the coupled-cluster singles and doubles method using a truncated pair natural orbital basis.[383]
- F. Neese, A. Hansen, F. Wennmohs, S. Grimme: Accurate Theoretical Chemistry with Coupled Electron Pair Models.[385]
- F. Neese, F. Wennmohs, A. Hansen: Efficient and accurate local approximations to coupled electron pair approaches. An attempt to revive the pair-natural orbital method.[398]
- D. G. Liakos, A. Hansen, F. Neese: Weak molecular interactions studied with parallel implementations of the local pair natural orbital coupled pair and coupled-cluster methods.[399]
- A. Hansen, D. G. Liakos, F. Neese: Efficient and accurate local single reference correlation methods for high-spin open-shell molecules using pair natural orbitals.[384]
- C. Riplinger, F. Neese: An efficient and near linear scaling pair natural orbital based local coupled-cluster method.[397]
- C. Riplinger, B. Sandhoefer, A. Hansen, F. Neese: Natural triple excitations in local coupled-cluster calculations with pair natural orbitals.[400]
- C. Riplinger, P. Pinski, U. Becker, E. F. Valeev, F. Neese: Sparse maps - A systematic infrastructure for reduced-scaling electronic structure methods. II. Linear scaling domain based pair natural orbital coupled cluster theory.[401]

- D. Datta, S. Kossmann, F. Neese: Analytic energy derivatives for the calculation of the first-order molecular properties using the domain-based local pair-natural orbital coupled-cluster theory[402]
- M. Saitow, U. Becker, C. Riplinger, E. F. Valeev, F. Neese: A new linear scaling, efficient and accurate, open-shell domain based pair natural orbital coupled cluster singles and doubles theory.[403]
- Y. Guo, C. Riplinger, U. Becker, D.G. Liakos, Y. Minenkov, L. Cavallo, and F. Neese: An improved linear scaling perturbative triples correction for the domain based local pair-natural orbital based singles and doubles coupled cluster method (DLPNO-CCSD(T)).[404]

Hence, it is sufficient to only point to a few significant design principles and features of these methods:

1. The correlation energy of any molecule can be written as a sum over the correlation energies of pairs of electrons, labelled by the internal indices (ij) of pairs of orbitals that are occupied in the reference determinant. If the orbitals (i) and (j) are localized, the pair correlation energy ϵ_{ij} falls off very quickly with distance, quite typically by about an order of magnitude per chemical bond that is separating orbitals (i) and (j). Hence, by using a suitable cut-off for a reasonable pair correlation energy estimate many electron pairs can be removed from the high-level treatment and only a linear scaling number of electron pairs will make a significant contribution to the correlation energy.
2. The natural estimate for the pair correlation energy comes from second order many body perturbation theory (MP2). However, canonical MP2 is scaling with the fifth power of the molecular size and hence, is not really attractive from a theoretical nor computational point of view. Owing to the small pre-factor RI-MP2 goes a long way to provide reasonably cheap estimates for pair correlation energies. However, if one uses localized internal orbitals, then the MP2 energy expression must be cast in form of linear equations. On the other hand, if one uses canonical virtual orbitals together with localized internal orbitals and neglects the coupling terms coming from purely internal Fock matrix elements F(i,k) and F(k,j) then one ends up with a fair approximation that is termed "semi-canonical MP2" in ORCA. It serves as a guess in the older LPNO method. For DLPNO this method is also not attractive.
3. In DLPNO, the guess is more sophisticated. Here the virtual space is spanned by projected atomic orbitals (PAOs) that are assigned to domains of atoms that are associated with each local internal orbital (i) and with the union of two such domains (i) and (j) for the electron pair (ij). If one applies the semi-local approximation, one obtains an excellent approximation to the semi-canonical MP2 energy. This is called the "semi-local" approximation and it scales linearly with respect to computational effort. If one iterates these equations to self-consistency to eliminate the coupling terms F(i,k) and F(k,j) then one obtains the full local MP2 method (LMP2 or L-MP2). By making the domains large enough the results approach the canonical MP2 energy to arbitrary accuracy while still being linear scaling with respect to computational resources. This method is the default for the DLPNO method.
4. Basically, the high-spin open-shell version of the DLPNO uses the same strategy as the closed-shell variant to efficiently generate the open-shell PNOs in a consistent manner to the closed-shell formalism. Through the development of the UHF-LPNO-CCSD method, we have realized that use of the unrestricted MP2 (UMP2) approach to define the open-shell PNOs introduces a few complexities; (1) the PNOs for β spin orbitals cannot be defined for α-α electron pairs and vice versa, (2) the diagonal PNOs for singly occupied orbitals cannot be properly defined, and (3) the PNO space does not become identical to that in the closed-shell LPNO framework in the closed-shell limit. However, to program all the unrestricted CCSD terms in the LPNO basis, those PNOs are certainly necessary. Therefore, in the UHF-LPNO-CCSD implementation, several terms, which, in many cases, give rather minor contributions in the correlation energy are omitted. Due to these facts, the UHF-LPNO-CCSD does not give identical results to that of RHF-LPNO-CCSD in the closed-shell limit. In addition, screening of the weak pairs on the basis of the semi-canonical UMP2 pair-energy results in somewhat unbalanced treatment of the closed- and open-shell states in some cases, leading to rather larger errors in the reaction energies. To overcome those issues, in the high-spin open-shell DLPNO-CCSD method, the PNOs are generated in the framework of semi-canonical NEVPT2 which smoothly converges to the RHF-MP2 counterpart in the closed-shell limit. The open-shell DLPNO-CCSD, which is made available from ORCA 4.0, includes a full set of the open-shell CCSD equations and is designed as a natural extension to the RHF-DLPNO-CCSD.
5. Screening of the electron pairs according to a truncation parameter (in ORCA it is called T_{CutPairs}) is not sufficient to obtain a highly performing local electronic structure method. The original work of Pulay suggested to limit excitations out of the internal orbitals (i) and (j) to the domain associated with the pair (ij). While this works well and has been implemented to perfection by Werner, Schütz and co-workers over the years,[405,

406, 407, 408] the pre-factor of such calculations is high, since the domains have to be chosen large in order to recover 99.9% or more of the canonical correlation energy.

6. The ORCA developers have therefore turned to an approach that has been used with a high degree of success in the early 1970s by Meyer, Kutzelnigg, Staemmler and their co-workers, namely the method of "pair natural orbitals" (PNOs).[409, 410, 411, 412]
   As shown by Loewdin in his seminal paper from 1955, natural orbitals (the eigenfunctions of the one-particle density matrix) provide the fastest possible convergence of the correlated wavefunction with respect to the number of one-particle functions included in the virtual space. It has been amply established that approximate natural orbitals are almost as succesful as the true natural orbitals (which would require the knowledge of the exact wavefunction) in this respect. While the success of approximate correlation treatments of many particle systems that use approximate natural orbitals of the whole systems are somewhat limited, this is not the case for pair natural orbitals. The latter have first been suggested as a basis for correlation calculations by England and co-workers and, at the time, were given the name "pseudonatural orbitals", a term that was used by Meyer throughout his pioneering work.

7. The PNOs are approximate natural orbitals of a given electron pair. In order to generate them one requires a one particle pair density matrix D_{ij} the eigenfunctions of which are the PNOs themselves while the corresponding eigenvalues are the PNO occupation numbers. While there are many creative possibilities that can lead to slightly different PNO sets, a quite useful and natural approximation is to generate such a density from the MP2 amplitudes as an expectation value (the "unrelaxed" MP2 density. One then uses a second threshold (in ORCAT_{CutPNO}) that controls the PNOs to be included in the calculation. PNOs with an occupation number $< T_{\text{CutPNO}}$ are neglected.

8. PNOs of a given electron pair form an orthonormal set. However, PNOs belonging to different electron pairs are not orthonormal and hence they overlap. This non-orthogonality leads to surprisingly few complications because the PNOs stay orthogonal to all occupied orbitals. In practice, the equations for PNO-based correlation calculations are hardly more complex than the canonical equations.

9. The nice feature of these pair densities is that they become small when the pair interaction becomes small. Hence weak pairs are correlated by very few PNOs. Therefore, the PNO expansion of the wavefunction is extremely compact and there only is a linear scaling number of significant excitation amplitudes that need to be considered.

10. A great feature of the PNOs is that they are "self-adapting" to the correlation situation that they are supposed to describe. Hence, they are as delocalized as required by the physics and there is no ad-hoc assumption about their location in space. However, it is clear that the PNOs are located in the same region of space as the internal orbitals that they correlate because otherwise they would not contribute to the correlation energy.

11. In the iterative local MP2, a set of PNOs is calculated for the MP2 calculation from the semicanonical amplitudes first using the cutoff `TCutPNO` ×`LMP2ScaleTCutPNO`. The size of the resulting PNO space should be comparable with DLPNO-MP2. After the iterations have converged, a (smaller) set of PNOs for the CCSD double excitation amplitudes is generated from the iterated local MP2 amplitudes in the (larger) PNO basis. The PNOs for the single excitation amplitudes are always calculated using the semicanonical MP2 amplitudes, as they require a much more conservative PNO truncation threshold.

12. Capitalizing on this feature one can define generous domains and expand the PNOs in terms of the PAOs and auxiliary fit functions (for the RI approximations) that are contained in these domains. In ORCA this is controlled by the third significant truncation parameter T_{CutMKN}. This is the basis of the DLPNO method. In the older LPNO method, the PNOs are expanded in terms of the canonical virtual orbitals and T_{CutMKN} is only used for a local fit to the PNOs. In the linear-scaling DLPNO implementation the domains expanding the PNOs in terms of the PAOs are controlled via T_{CutDO}.

13. PNO expansions have the amazing advantage that the PNOs converge to a well-defined set as the basis set is approaching completeness. Hence, the increase in the number of PNOs per electron pair is very small upon enlargement of the orbital basis. In other words, correlation calculations with large basis sets are not that much more expensive than correlation calculations with small basis sets. Thus, the advantage of PNO over canonical calculations increases with the size of the basis set. This is a great feature as large and flexible basis sets are a requirement for meaningful correlation calculations.

14. In summary, DLPNO and LPNO calculations are controlled by only three cut-off parameters with well-defined meanings: a) T_{CutPairs}, the cut-off for the electron pairs to be included in the coupled-pair or coupled-cluster iterations, b) T_{CutPNO} which controls how many PNOs are retained for a given electron pair and c) T_{CutMKN} that

controls how large the domains are that the PNOs expand over. For the linear-scaling DLPNO calculations the domain sizes are controlled via T_{CutDO}.

15. It is clear that owing to the truncations a certain amount of error is introduced in the results. However, having the LMP2 results available, one can compensate for the errors coming from T_{CutPairs} and T_{CutPNO}. This is done in ORCA and the correction is added to the final correlation energy, thus bringing it very close (to mEh accuracy or better) to the canonical result. T_{CutMKN} is best dealt with by making it conservative (at 1e−3 to 1e−4 the domains are about 20–30 atoms large, which is sufficient for an accurate treatment).

16. Note that the LPNO and DLPNO methods do not introduce any real space cut-offs. We refrain from doing so and insist in our method development by making all truncations based on wavefunction or energy parameters. We feel that this is the most unbiased approach and it involves no element of "chemical intuition" or "prejudice".

17. In the DLPNO method a highly efficient screening mechanism is operative. In this method one first obtains a (quadratically scaling) multipole estimate for the pair correlation energy that is extremely fast to compute (a few seconds, even for entire proteins). Only if this estimate is large enough, a given electron pair is even considered for a LMP2 treatment. Quite typically pairs with energy contributions of 1e−6 Eh and smaller are very well described by the dipole-dipole estimate. Specifically, we drop pairs with estimated energies $< 0.01 \times T_{\text{CutPairs}}$ and add their multipole energy sum to the final correlation energy. These corrections tend to be extremely small, even for large molecules and are insignificant for the energy. However, importantly, the multipole estimate ensures linear scaling in the MP2 treatment. The pairs that then do not survive the pair-prescreening are called "weak pairs" in the ORCA or DLPNO sense. They still play a role in the calculation of the triple excitation correction.

18. The calculation of triple excitation contributions is more involved and one does not have a perturbative estimate available since the (T) contribution is perturbative itself. While the (T) contribution is much smaller than the CCSD correlation energy, the errors introduced by the various local and PNO approximations can be significant. We found that one has to include triples with at least one pair being a "weak" LMP2 pair (with its LMP2 amplitudes) in order to arrive at sufficiently accurate results.

Basic Usage

The use of the DLPNO methods is simple and requires little special attention from the user:

```
# Local Pair Natural Orbital Test
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD TightSCF
%maxcore 2000

# these are the default values - they need not to be touched!
%mdci  TCutPNO   3.33e-7   # cutoff for PNO occupation numbers. This
                             is the main truncation parameter
       TCutPairs 1e-4  # cut-off for estimated pair correlation energies.
                         This exploits the locality in the internal space
       TCutMKN   1e-3  # this is a technical parameter here that controls the␣
↪domain
                         size for the local fit to the PNOs. It is conservative.
       end

* xyz 0 1
... (coordinates)
*
```

The LPNO methods used to run in strict analogy to their canonical variants. In fact, one should not view the LPNO methods as new model chemistry - they are designed to reproduce the canonical results, including BSSE. This is different from the domain based local correlation methods that do constitute a new model chemistry with properties that are different from the original methods.

In some situations, it may be appropriate to adapt the accuracy of the calculation. Sensible defaults have been determined from extensive benchmark calculations and are accessible via LoosePNO, NormalPNO and TightPNO keywords in the simple input line.[413]

These keywords represent the recommended way to control the accuracy of DLPNO calculations as follows. Manual changing of thresholds beyond these specifying these keywords is usually discouraged.

```
# Tight settings for increased accuracy, e.g. when investigating
# weak interactions or conformational equilibria
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD(T) TightPNO TightSCF

# OR: Default settings (no need to give NormalPNO explicitly)
# Useful for general thermochemistry
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD(T) NormalPNO TightSCF

# OR: Loose settings for rapid estimates
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD(T) LoosePNO TightSCF

%maxcore 2000

* xyz 0 1
... (coordinates)
*
```

Since ORCA 4.0, the linear-scaling DLPNO implementation described in reference [401] is the default DLPNO algorithm. However, for comparison, the first DLPNO implementation from references [397] and [400] can still be called by using the DLPNO2013 prefix instead of the DLPNO- prefix.

```
# DLPNO-CCSD(T) calculation using the 2013 implementation
! cc-pVTZ cc-pVTZ/C DLPNO2013-CCSD(T)

# DLPNO-CCSD(T) calculation using the linear-scaling implementation
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD(T)

* xyz 0 1
... (coordinates)
*
```

Until ORCA 4.0, the "semi-canonical" approximation is used in the perturbative triples correction for DLPNO-CCSD. It was found that the "semi-canonical" approximation is a very good approximation for most systems. However, the "semi-canonical" approximation can introduce large errors in rare cases (particularly when the HOMO-LUMO gap is small), whereas the DLPNO-CCSD is still very accurate. To improve the accuracy of perturbative triples correction, since 4.1, an improved perturbative triples correction for DLPNO-CCSD is available, DLPNO-CCSD(T1)[404]. In DLPNO-CCSD(T1), the triples amplitudes are computed iteratively, which can reproduce more accurately the canonical (T) energies.

It is necessary to clarify the nomenclature used in ORCA input files. The keyword to invoke "semi-canonical" perturbative triples correction approximation is DLPNO-CCSD(T). While, the keyword of improved iterative approximation is DLPNO-CCSD(T1). However, in our recent paper[404], the "semi-canonical" perturbative triples correction approximation is named DLPNO-CCSD(T0), whereas the improved iterative one is called DLPNO-CCSD(T). Thus, the names used in our paper are different from those in ORCA input files. An example input file to perform improved iterative perturbative triples correction for DLPNO-CCSD is given below,

```
# DLPNO-CCSD(T1) calculation using the iterative triples correction
! cc-pVTZ cc-pVTZ/C DLPNO-CCSD(T1)

%mdci
      TNOSCALES    10.0 #TNO truncation scale for strong triples,␣
↪TNOSCALES*TCutTNO.
                        Default setting is 10.0
      TNOSCALEW   100.0 #TNO truncation scale for weak triples, TNOSCALEW*TCutTNO
                        Default setting is 100.0
      TriTolE      1e-4 #(T) energy convergence tolerance
                        Default setting is 1e-4
%end
* xyz 0 1
```

(continues on next page)

(continued from previous page)

```
... (coordinates)
*
```

Since ORCA 4.2, the improved iterative perturbative triples correction for open-shell DLPNO-CCSD is available as well. The keyword of open-shell DLPNO-CCSD(T) is the same as that of the closed-shell case.

Since ORCA 4.0, the high-spin open-shell version of the DLPNO-CISD/QCISD/CCSD implementations have been made available on top of the same machinery as the 2016 version of the RHF-DLPNO-CCSD code. The present UHF-DLPNO-CCSD is designed to be an heir to the UHF-LPNO-CCSD and serves as a natural extension to the RHF-DLPNO-CCSD. A striking difference between UHF-LPNO and newly developed UHF-DLPNO methods is that the UHF-DLPNO approach gives identical results to that of the RHF variant when applied to closed-shell species while UHF-LPNO does not. Usage of this program is quite straightforward and shown below:

```
# (1) In case of ROHF reference
! ROHF DLPNO-CCSD def2-TZVPP def2-TZVPP/C TightSCF TightPNO

# (2) In case of UHF reference, the QROs are constructed first and used for
#     the open-shell DLPNO-CCSD computations
! UHF DLPNO-CCSD def2-TZVPP def2-TZVPP/C TightSCF TightPNO

# (3) In case that UKS is specified, the QROs are constructed first and used as
#     "unconverged" UHF orbitals for the open-shell DLPNO-CCSD computations.
#     This approach is useful when the converged UHF wavefunction is qualitatively
#     wrong but the UKS wavefunction is not
! UKS CAM-B3LYP DLPNO-CCSD def2-TZVPP def2-TZVPP/C TightSCF TightPNO
```

Note

DLPNO-CISD/QCISD/CCSD methods are dedicated to closed-shell and high-spin open-shell species, but not spin-polarized systems (e.g. open shell singlets or antiferromagnetically coupled transition metal clusters). Performing DLPNO-CISD/QCISD/CCSD calculations upon open shell singlet UHF/UKS wavefunctions will give results resembling the corresponding closed shell singlet calculations, because the DLPNO calculations will be done on the closed-shell determinant composed of the QRO orbitals. Similarly, calculations of spin-polarized systems other than open shell singlets may give qualitatively wrong results. Here, the UHF-LPNO-CCSD method or the DLPNO-NEVPT2 are better alternatives.

The same set of truncation parameters as closed-shell DLPNO-CCSD is used also in case of open-shell DLPNO. The open-shell DLPNO-CCSD produces more than 99.9 % of the canonical CCSD correlation energy as in case of the closed-shell variant. The computational timings of the UHF-DLPNO-CCSD and RIJCOSX-UHF for linear alkane chains in triplet state are shown in Fig. 3.10.

Fig. 3.10: Computational times of RIJCOSX-UHF and UHF-DLPNO-CCSD for the linear alkane chains ($C_n H_{2n+2}$) in triplet state with def2-TZVPP basis and default frozen core settings. 4 CPU cores and 128 GB of memory were used on a single cluster node.

Although those systems are somewhat idealized for the DLPNO method to best perform, it is clear that the preceding RIJCOSX-UHF is the rate-determining step in the total computational time for large examples. In the open-shell DLPNO implementations, SOMOs are included not only in the occupied space but also in the PNO space in the preceding integral transformation step. This means the presence of more SOMOs may lead to more demanding PNO integral transformation and DLPNO-CCSD iterations. The illustrative examples include active site model of the [NiFe] Hydrogenase in triplet state and the oxygen evolving complex (OEC) in the high-spin state, which are shown in Figures *7* and *8*, respectively. With def2-TZVPP basis set and NormalPNO settings, a single point calculation on [NiFe] Hydrogenase (Fig. 3.11) took approximately 45 hours on a single cluster node by using 4 CPU cores of Xeon E5-2670. A single point calculation on the OEC compound (Fig. 3.12) with the same computational settings finished in 44 hours even though the number of AOs in this system is even fewer than the Hydrogenase: the Hydrogenase active site model and OEC involve 4007 and 2606 AO basis functions, respectively. Special care should be taken if the system possesses more than ten SOMOs, since inclusion of more SOMOs may drastically increase the prefactor of the calculations. In addition, if the SOMOs are distributed over the entire molecular skeleton, each pair domain may not be truncated at all; in this case speedup attributed to the domain truncation will not be achieved at all.

Fig. 3.11: Ni-Fe active center in the [NiFe] Hydrogenase in its second-coordination sphere. The whole model system is composed of 180 atoms.

Fig. 3.12: A model compound for the OEC in the S_2 state of photosystem II which is composed of 238 atoms. In its high-spin state, the OEC possesses 13 SOMOs in total.

Calculation of the orbital-unrelaxed density has been implemented for closed-shell DLPNO-CCSD. This permits analytical computation of first-order properties, such as multipole moments or electric field gradients. In order to reproduce conventional unrelaxed CCSD properties to a high degree of accuracy, tighter thresholds may be needed than given by the default settings. Reading of the reference[402] is recommended. Calculation of the unrelaxed density is requested as usual:

```
%MDCI Density Unrelaxed End
```

There are a few things to be noticed about (D)LPNO methods:

- The DLPNO and LPNO methods obligatorily make use of the RI approximation. Hence, a correlation fit set must be provided.
- The DLPNO-CCSD(T) method is applicable to closed-shell or high-spin open-shell species. When performing DLPNO calculations on open-shell species, it is always better to have UCO option: If preceding SCF converges to broken-symmetry solutions, it is not guaranteed that the DLPNO-CCSD gives physically meaningful results.
- The open-shell version of the DLPNO approach uses a different strategy to the LPNO variant to define the

open-shell PNOs. This ensures that, unlike the open-shell LPNO, the PNO space converges to the closed-shell counterpart in the closed-shell limit. Therefore, in the closed-shell limit, the open-shell DLPNO gives identical correlation energy to the RHF variant up to at least the third decimal place. The perturbative triples correction referred to as, (T), is also available for the open-shell species.

- When performing a calculation on the open-shell species with either of canonical/DLPNO methods on top of the Slater determinant constructed from the QROs, special attention should be paid on the orbitals energies of those QROs. In some cases, the orbitals energy of the highest SOMO appear to be higher than that of the lowest VMO. Similarly to this, the orbital energy of the highest DOMO may appear to higher than that of the lowest SOMOs. In such cases, the CEPA/QCISD/CCSD iteration may show difficulty in convergence. In the worst case, it just diverges. Most likely, in such cases, one has to suspect the charge and multiplicity might be wrong. If they are correct, you may need much prettier starting orbitals and a bit of good luck! Apart from a careful choice of starting orbitals (in particular, DFT orbitals can be used in place of the default HF orbitals if the latter have qualitative deficiencies, including but not limited to severe spin contamination), changing the maximum DIIS expansion space size (`MaxDIIS`) and the level shift (`LShift`) in the `%mdci` block may alleviate the convergence problems to some extent.
- DLPNO-CCSD(T)-F12 and DLPNO-CCSD(T1)-F12 (iterative triples) are available for both closed- and open-shell cases. These methods employ a perturbative F12 correction on top of the DLPNO-CCSD(T) correlation energy calculation. The F12 part of the code uses the RI approximation in the same spirit as the canonical RI-F12 methods (refer to section *Explicitly Correlated Methods: F12-MP2 and F12-CCSD(T)*). Hence, they should be compared with methods using the RI approximation for both CC and F12 parts. The F12 correction takes only a fraction (usually 10-30%) of the total time (excluding SCF) required to calculate the DLPNO-CCSD(T)-F12 correlation energy. Thus, the F12 correction scales the same (linear or near-linear) as the parent DLPNO method. Furthermore, no new truncation parameters are introduced for the F12 procedure, preserving the black-box nature of the DLPNO method. The F12D approximation is highly recommended as it is computationally cheaper than the F12 approach which involves a double RI summation. Keywords: DLPNO-CCSD(T)-F12D, DLPNO-CCSD(T)-F12, DLPNO-CCSD(T1)-F12D, DLPNO-CCSD(T1)-F12, DLPNO-CCSD-F12D, DLPNO-CCSD-F12.
- There are three thresholds that can be user controlled that can all be adjusted in the %mdci block: (a) T_{CutPNO} controls the number of PNOs per electron pair. This is the most critical parameter and has a default value of 3.33×10^{-7}. (b) T_{CutPairs} controls a perturbative selection of significant pairs and has a default value of 10^{-4}. (c) T_{CutMKN} is a technical parameter and controls the size of the fit set for each electron pair. It has a default value of 10^{-3}. All of these default values are conservative. Hence, no adjustment of these parameters is necessary. All DLPNO-CCSD truncations are bound to these three truncation parameters and should almost not be touched (Hence they are also not documented :)).
- The preferred way to adjust accuracy when needed is to use the "LoosePNO/NormalPNO/TightPNO" keywords. In addition, "TightPNO" triggers the full iterative (DLPNO-MP2) treatment in the MP2 guess, whereas the other options use a semicanonical MP2 calculation. Table 3.40 and Table 3.41 contain the thresholds used by the current (2016) and old (2013) implementations, respectively.
- Potential energy surfaces are virtually but not perfectly smooth (like any method that involves cut-offs). Numerical gradient calculations have been attempted and reported to have been successful.
- The DLPNO methods do work together with RIJCOSX, RI-JK and also with ANO basis sets and basis set extrapolation. They also work for conventional integral handling.
- The methods behave excellently with large basis sets. Thus, they stay efficient even when large basis sets are used that are necessary to obtain accurate results with wavefunction based *ab initio* methods. This is a prerequisite for efficient computational chemistry applications.
- For LPNO-CCSD, calculations with about 1000 basis functions are routine, calculations with about 1500 basis functions are possible and calculations with 2000-2500 basis functions are the limit on powerful computers. For DLPNO-CCSD much larger calculations are possible. There is virtually no crossover and DLPNO-CCSD is essentially always more efficient than LPNO-CCSD. Starting from about 50 atoms the differences become large. The largest DLPNO-CCSD calculation to date featured >1000 atoms and more than 20000 basis functions!
- Using large main memory is not mandatory but advantageous since it speeds up the initial integral transformation significantly (controlled by "MaxCore" in the %mdci block, see section *Local correlation (DLPNO)*).
- The open-shell versions are about twice as expensive as the corresponding closed-shell versions.

- Analytic gradients are not available.
- An unrelaxed density implementation is available for closed-shell DLPNO-CCSD, permitting calculation of first-order properties.

Table 3.40: Accuracy settings for DLPNO coupled cluster (current version).

Setting	T_{CutPairs}	T_{CutDO}	T_{CutPNO}	T_{CutMKN}	MP2 pair treatment
LoosePNO	10^{-3}	2×10^{-2}	1.00×10^{-6}	10^{-3}	semicanonical
NormalPNO	10^{-4}	1×10^{-2}	3.33×10^{-7}	10^{-3}	semicanonical
TightPNO	10^{-5}	5×10^{-3}	1.00×10^{-7}	10^{-3}	full iterative

Table 3.41: Accuracy settings for DLPNO coupled cluster (deprecated 2013 version).

Setting	T_{CutPairs}	T_{CutPNO}	T_{CutMKN}	MP2 pair treatment
LoosePNO	10^{-3}	1.00×10^{-6}	10^{-3}	semicanonical
NormalPNO	10^{-4}	3.33×10^{-7}	10^{-3}	semicanonical
TightPNO	10^{-5}	1.00×10^{-7}	10^{-4}	full iterative

As an example, see the following isomerization reaction that appears to be particularly difficult for DFT:

Isomerizes to:

The results of the calculations (closed-shell versions) with the def2-TZVP basis set (about 240 basis functions) are shown below:

Method	Energy Difference (kcal/mol)	Time (min)
CCSD(T)	-14.6	92.4
CCSD	-18.0	55.3
LPNO-CCSD	-18.6	20.0
CEPA/1	-12.4	42.2
LPNO-CEPA/1	-13.5	13.4

The calculations are typical in the sense that: (a) the LPNO methods provide answers that are within 1 kcal/mol of the canonical results, (b) CEPA approximates CCSD(T) more closely than CCSD. The speedups of a factor of 2 – 5 are moderate in this case. However, this is also a fairly small calculation. For larger systems, speedups of the LPNO methods compared to their canonical counterparts are on the order of a factor >100–1000.

Keywords

Given these explanations the various cut-off parameters that can be controlled in LPNO and DLPNO calculations should be understandable and are listed below. We emphasize again that only the three thresholds T_{CutPairs}, T_{CutPNO} and T_{CutMKN} should be touched by the user, unless very specific questions are addressed. The recommended way to control the accuracy of calculations is to specify "TightPNO", "NormalPNO" or "LoosePNO" keywords, rather than to change numeric values of cutoffs. Individual thresholds should normally not be changed, as the defaults are sensible and lead to good cost/performance ratios.

```
%mdci TCutPairs        1e-4   # cut-off for the pair truncation
      TCutPNO       3.33e-7   # cut-off for the PNO truncation
      TCutDO           1e-2   # cut-off for the DLPNO domain construction
      TCutMKN          1e-3   # cut-off for the local fit
                              # for DLPNO2013: also domain construction
                              # remaining options, tied to the three main cut-offs,
                                EXPERTS ONLY!
      Localize         true   # flag for using localized orbitals
      LocMet           AHFB   # Localization method.
                              # Options: PM, FB, IAOIBO, IAOBOYS, NEWBOYS, AHFB
      LocTol           1e-6   # Absolute threshold for the localization procedure
                              # Automatically adjusted by default.
      LocTolRel        1e-8   # Relative threshold for the localization procedure
      LocMaxIter        128   # Maximum number of localization iterations
      LocRandom           1   # default, take random seed for any localization
                              # For internal orbitals: choose best of 32␣
↪localizations
                              # Switched off for AHFB
                          0   # take constant seed for any localization (for␣
↪testing)
      LocNAttempts       np   # number of localization attempts
                              # default: number of processes, minimum 8, if
                              #  randomize true
                              # 1, if randomize false
                              # any number larger or equal np, if randomize true

      PNONorm       MP2Norm   # default, old IEPANorm can also be used
                                (near identical results)
      NrMP2Pairs_Trip     1   # number of MP2 pairs to be included in the triples
                                calculation
      PAOOverlapThresh 1e-8   # generation of non-redundant PAOs from redundant␣
↪ones
      UseFullLMP2Guess false  # Use iterative full LMP2 (for DLPNO)
      SinglesFockUsePNOs true # compute the Singles Fock matrix (SFM) in PNOs.
                              # DLPNO2013: default for SinglesFockUsePNOs is false,
                              #    by default RIJCOSX is used for the SFM, except␣
↪when
                              #    RIJK is given. In that case the RIJK-SFM is␣
↪used.
      LMP2MaxIter        50   # max no of iterations in the MP2 equations
      LMP2TolE         1e-7   # LMP2 energy convergence tolerance
      LMP2TolR         5e-7   # LMP2 residual convergence tolerance
      LMP2ScaleTCutPNO        # PNO cutoff for LMP2 is: TCutPNO*LMP2ScaleTCutPNO
                              # Default: TCutPNO(DLPNO-MP2)/TCutPNO(DLPNO-CCSD)␣
↪with
                              # respective TCutPNOs specific to Loose/Normal/
↪TightPNO
```

(continues on next page)

(continued from previous page)

```
      LMP2FCut         1e-5   # LMP2 neglect cut-off for
                                off-diagonal Fock matrix elements
      TCutTNO             0   # Cut-off for triples natural orbitals (0=automatic)
      TCutPNOSingles     -1   # -1= use 0.03*TCutPNO
      TCutPreScr         -1   # -1= use 0.01*TCutPairs for
                                multipole estimate based screening
      TCutDeloc         0.1   # delocalization threshold for specification
                                of extended domains.
                                Necessary because PAOs are not strictly localized
      TCutOSV          1e-6   # orbital specific virtuals used for pre-screening.
                                No critical
      TScaleMP2Pairs    0.1
      TScaleMKNStrong    10
      TScaleMKNWeak     100
```

For larger systems and tighter thresholds the disk I/O of a DLPNO calculation may become challenging. In this case, it might be usefull to keep some integrals in memory, if enough RAM is available. With the following flag

```
%mdci
     StorageType Shared
end
```

ORCA will try to store certain much-used integrals in shared memory. If the amount of memory is not sufficient, ORCA will fall back to on-disk storage. **NOTE:** This flag will only work if all processes work on the same node.

IMPORTANT NOTE REGARDING ORBITAL LOCALIZATION

- **Localized orbitals for DLPNO are obtained via the Foster-Boys method with an augmented Hessian converger (`AHFB`) by default.**
- The localization tolerance (`LocTol`) is coupled to the SCF gradient tolerance (`TolG`) with a constant factor by default. Selecting specific SCF convergence settings (such as `TightSCF`) therefore also ensures obtaining a set of appropriately well converged localized orbitals. This can be overridden by setting a different value for `LocTol`.
- An important feature of the augmented Hessian converger is that it systematically approaches a local maximum of the localization function (even though convergence to the global maximum cannot be guaranteed). As opposed to that, the conventional localization method (`FB`) may stop, for example, at a saddle point. In bad cases, this can lead to deviations of several kJ/mol in the DLPNO energy. Likewise, it can contribute towards lack of reproducibility of results.
- No similar procedure has been implemented for the other localization methods (such as Pipek-Mezey) yet. The same problems as with the FB converger can occur in these cases.
- No randomization is used for the `AHFB` converger.

The old default orbital localization settings of ORCA 4.0 can be reproduced with the following options:

```
%MDCI
  LocMet    FB
  LocTol    1.0e-6
  LocRandom 1
End
```

Regarding the methods that employ randomization (`FB`, `PM`, `IAOIBO`, `IAOBOYS`) only, the following notes apply:

- Generally, better DLPNO results are obtained when several runs of localization are undertaken using different initial guesses. The different initial guesses are obtained using randomization (LocRandom).
- However, randomization of the initial guess can lead to differently localized MOs in different calculations. This can yield non-identical correlation energies, varying in the sub-kJ/mol range, for different runs on the same machine.

- In order to yield identical correlation energy results, randomization can be switched off (LocRandom 0). However, switching off randomization only leads to identical results on the same machine, but can still lead to slightly different results (in the sub-kJ/mol range) on different machines.
- Reproducibility of the correlation energy is expected to increase further if LocNAttempts is set to higher values.

The input below shows how to perform a DLPNO calculation with settings that exactly reproduce the canonical RI-MP2 result. They are not recommended for production use, but merely to show that if the local approximations are pushed, then the result coincides with the canonical one. If one would set T_{CutPNO} to zero this would give canonical RI-CCSD. However, this is an absurdly inefficient calculation and hence not done.

```
#
! def2-SV(P) def2/JK def2-SVP/C RI-JK DLPNO-CCSD VeryTightSCF RI-MP2

# obtain a result that only contains errors from the PNO approximation
# but no others
%mdci TCutPairs        0
      TCutMKN          0
      UseFullLMP2Guess true
      LMP2FCut         1e-9
      LMP2MaxIter      25
      LMP2TolE         1e-10
      LMP2TolR         1e-11
      PAOOVerlapThresh 1e-9
      end

! Bohrs
* xyz 0 1
   C  -1.505246952209632    1.048213673267046   -3.005665895986369
   C   1.289678561934891    0.246429688933291   -3.259735682020124
   C   2.834670835163566    1.157307360133605   -0.990383454919828
   C   1.924119415395082   -0.128330938291771    1.465070676514038
   C  -0.931529472233802   -0.722841293992075    1.397639867298547
   C  -2.347670084056626    1.213332291655600   -0.217984867773136
   H   2.084955694093313    0.973408301535989   -5.037750251258102
   H   1.426532559234904   -1.831017720289521   -3.371063003813707
   H  -1.795307501459984    2.891278294563413   -3.927855043896308
   H  -2.709613973668925   -0.308515546176734   -4.026627646697411
   H  -4.404246093821399    0.941639912907262   -0.071175054238094
   H  -1.962867323232915    3.122079490952855    0.528101313545138
   H  -1.245096579039474   -2.621186110634707    0.594784162223769
   H  -1.699155144887690   -0.782162821007662    3.328959985756973
   H   2.347109421287126    1.104305785540561    3.087624818244846
   H   2.990679065503112   -1.888017241218143    1.775287572161196
   H   4.862301668284708    0.796425411350593   -1.279131939569907
   H   2.634027658640572    3.226752635113244   -0.827936424652650
*
```

Including (semi)core orbitals in the correlation treatment

In some chemical applications some or all of the chemical (semi)core electrons must be included in the correlation treatment. In this case, it is necessary to tighten the TCutPNO thresholds for electron pairs in which chemical (semi)core electrons are involved. This is now the default in DLPNO calculations.

For instance, one can decide to switch off the frozen-core approximation and include all the electrons in the correlation treatment. In this case, the program will use tighter thresholds by default for all electron pairs and Singles that involve chemical core electrons. Note that, in this case, the use of properly optimized basis functions for correlating the inner electrons is highly recommened.

```
! DLPNO-CCSD(T) def2-SVP def2-SVP/C NoFrozenCore
%mdci TSCALEPNO_CORE 0.01 # scaling factor for TCutPNO for electron pairs and
```

(continues on next page)

(continued from previous page)

```
                          # Singles involving chemical core electrons
      end
* xyz 0 1
   Ti   0.0001595288      0.0000775041      0.0000000000
   F    1.7595996122      0.0000634675     -0.0000000011
   F   -0.5865076471      1.6586935196      0.0000000018
   F   -0.5866248292     -0.8294172469     -1.4362516915
   F   -0.5866248311     -0.8294172443      1.4362516907
*
```

Another option is to choose the involved chemical core electrons by using an energy window. In this way all electron pairs and Singles that involve chemical core electrons, which are in the defined energy window, are affected by TScalePNO_CORE.

```
! DLPNO-CCSD(T) def2-SVP def2-SVP/C
%method
  FrozenCore  FC_EWIN
end
%mdci
  EWIN -40, 10000
end
* xyz 0 1
   Ti   0.0001595288      0.0000775041      0.0000000000
   F    1.7595996122      0.0000634675     -0.0000000011
   F   -0.5865076471      1.6586935196      0.0000000018
   F   -0.5866248292     -0.8294172469     -1.4362516915
   F   -0.5866248311     -0.8294172443      1.4362516907
*
```

A summary with the number of electrons affected by TScalePNO_Core for the examples just discussed is shown in Table Table 3.42.

Table 3.42: Number of chemical core electrons included in the DLPNO calculation and affected by TScalePNO_Core for the TiF_4 examples

Keyword	Chemical Core Electrons		Valence Electrons
	Frozen	Included[a]	
FrozenCore (default)	18	0	40
NoFrozenCore	0	18	40
EWIN -40, 10000	16	2	40

[a] using TScalePNO_Core.

By default, ORCA provides a chemical meaningful definition for the number of electrons which belong to the chemical core of each element. As already discussed, these default values define which pairs are affected by TScalePNO_Core. However, the user can modify the number of chemical core electrons for a specific element via the NewNCore keyword.

```
! DLPNO-CCSD(T) def2-SVP def2-SVP/C NoFrozenCore
%method
   NewNCore Ti 8 end
end
* xyz 0 1
   Ti   0.0001595288      0.0000775041      0.0000000000
   F    1.7595996122      0.0000634675     -0.0000000011
```

(continues on next page)

(continued from previous page)

```
   F   -0.5865076471      1.6586935196      0.0000000018
   F   -0.5866248292     -0.8294172469     -1.4362516915
   F   -0.5866248311     -0.8294172443      1.4362516907
*
```

In the previous example, the number of chemical core electrons for Ti has been fixed to 8.

Starting from ORCA 6.0, in DLPNO calculations, tightened TCutPNO thresholds are used by default for "semicore" electron pairs involving the 3s and 3p orbitals of first-row transition metals.[414] This improves not only the accuracy of the results noticeably but also the efficiency of the computations as the system size grows (ref.). To reproduce results obtained with earlier versions, the number of semicore orbitals on first-row transition metals needs to be set to zero (the old default) instead of eight (the current default), as done below for a Zn atom with the NewNSemiCore keyword in the method block:

```
%method NewNSemiCore Zn 0 end end
```

Note

- Of course, if electrons are replaced by ECPs, they are not included in the correlation treatment.
- If ECPs are used, the number for NewNCore has to include the electrons represented by the ECPs as well. E.g. if an element is supposed to have 60 electrons in the ECP and additional 8 electrons should be frozen in the correlation calculation, NewNCore should be 68.
- The different sets of orbitals (chemical core electrons included in the correlation treatment and valence electrons) are localized separately in order to avoid the mixing of core and valence orbitals.

3.10.11 Multi-Level Calculations

In many applications events are investigated that are located in a relatively small region of the system of interest. In these cases, combined quantum-mechanics/molecular-mechanics (QM/MM) approaches have been proved to be extremely useful, especially in the modeling of enzymatic reactions. The basic idea of any QM/MM method is to treat a small region of the system at the QM level and to use an MM treatment for the remaining part of the system. Alternatively, QM/QM methods, where different parts of a system are studied at various quantum mechanical levels, are also available. Quantum mechanical methods are more computationally demanding than the molecular mechanics treatment, and this limits the applicability of all-QM approaches. Nevertheless, QM/QM methods retain some strong advantages over QM/MM schemes. For instance, force field parameters for the molecular mechanics part of the calculation are not necessary, and thus there are no restrictions on the type of chemical systems that can be treated. Moreover, problems usually caused by boundaries between QM and MM parts do not occur. Finally, the accuracy of an all-QM calculation is expected to be higher compared to the accuracy of QM/MM approaches, that is limited by the MM treatment.

In ORCA, the different methods that can be used in a QM/QM calculations are:

- DLPNO-CCSD(T) with TightPNO thresholds
- DLPNO-CCSD(T) with NormalPNO thresholds
- DLPNO-CCSD(T) with LoosePNO thresholds
- DLPNO-CCSD
- DLPNO-MP2
- HF

The user can define an arbitrary number of *fragments* in the input, the level of theory to be used for each fragment and for the interaction between different fragments . Localized molecular orbitals are then assigned to a given fragment on the basis of their Mulliken populations.

The following example shows the calculation of a benzene dimer, for which the individual monomers are calculated on MP2 level, and the interaction between the two monomers is calculated on TightPNO DLPNO-CCSD(T) level. More realistic use cases are discussed in ref. [355].

```
! DLPNO-CCSD(T) Def2-SVP Def2-SVP/C
%mdci
  UseFullLmp2Guess True
  TightPNOFragInter  {1 2}
  MP2FragInter  {1 1} {2 2}
end

*xyz 0 1
C(1)       1.393      0.000      0.0
H(1)       2.474      0.000      0.0
C(1)       0.695      1.206      0.0
H(1)       1.238      2.143      0.0
C(1)      -0.695      1.206      0.0
H(1)      -1.238      2.143      0.0
C(1)      -1.393      0.000      0.0
H(1)      -2.474      0.000      0.0
C(1)      -0.695     -1.206      0.0
H(1)      -1.238     -2.143      0.0
C(1)       0.695     -1.206      0.0
H(1)       1.238     -2.143      0.0
C(2)       2.333      1.33       3.5
H(2)       3.414      1.33       3.5
C(2)       1.635      2.536      3.5
H(2)       2.178      3.473      3.5
C(2)       0.245      2.536      3.5
H(2)      -0.298      3.473      3.5
C(2)      -0.453      1.33       3.5
H(2)      -1.534      1.33       3.5
C(2)       0.245      0.124      3.5
H(2)      -0.298     -0.813      3.5
C(2)       1.635      0.124      3.5
H(2)       2.178     -0.813      3.5
*
```

- For the calculation of the interaction energy, the energy of the individual benzene monomer should be calculated on the accuracy level of the monomer in the dimer calculation, i.e. using MP2 with full LMP2 guess for the above example.

All possible settings for the multi-level calculation are listed below.

```
# The one-keyword line defines the default method for the multi-level calculation.
# Options here are DLPNO-CCSD(T) or DLPNO-CCSD with the addition of the
# LoosePNO, NormalPNO and TightPNO keyword
!DLPNO-CCSD(T)

# The below given keywords define the changes with respect to the
# above given default method. The user should take care that each intra- or
# interfragment combination is defined only once (unlike in the example given␣
↪below)

%mdci
  LoosePNOFragInter {1 1} {2 2}       # use LoosePNO settings for the intrafragment
                                      # pair energies of fragments 1 and 2
  NormalPNOFragInter {1 2}            # use NormalPNO settings for the␣
↪interfragment
                                      # pair energies between fragment 1 and 2
  TightPNOFragInter {1 3}             # use TightPNO settings for the interfragment
                                      # pair energies between fragment 1 and 3
  NormalPNOTightPairFragInter {1 2}   # use NormalPNO settings but with TCutPairs
```

(continues on next page)

(continued from previous page)

```
                                        # 1.e-5 for the interfragment pair energies
                                        # between fragment 1 and 2
  LoosePNOTightPairFragInter {1 3}      # use LoosePNO settings but with TCutPairs 1.
↪e-5
                                        # for the interfragment pair energies between
                                        # fragment 1 and 3
  NoTriplesFragments  1, 2              # if all MOs of a triple are located on␣
↪fragment
                                        # 1 and / or 2, the triple is neglected
  MP2FragInter  {1 1} {2 2}             # compute the intrafragment pair energies of
                                        # fragments 1 and 2 on MP2 level
  HFFragInter  {1 1} {2 2}              # compute the intrafragment energies on HF␣
↪level
  UseFullLmp2Guess false                # default = false,
                                        # if MP2FragInter is used: default = true
end

*xyz 0 1
C(1)       1.393      0.000      0.0
H(1)       2.474      0.000      0.0
...
```

3.10.12 Multi-Level Calculations for IP and EA-EOM-DLPNO-CCSD

The multi-layer method can be used to include the environmental effect in IP-and EA-EOM-DLPNO-CCSD method. A typical input file for the multi-layer IP-EOM-CCSD method will look like

```
! IP-EOM-DLPNO-CCSD TightSCF NORMALPNO def2-SVP def2-SVP/C def2/J RIJCOSX

%mdci
nroots 4
DTOl 1e-7
NORMALPNOFragInter { 1 1}
LOOSEPNOFragInter { 1 1}
HFFRAGMENTINTERACTION { 2 2}
end

*xyz 0 1
C(1)       2.782064    -1.456235     0.000000
C(1)       1.478695    -0.729491     0.000000
C(1)       0.274461    -1.343436     0.000000
N(1)      -0.914790    -0.659079     0.000000
C(1)      -0.988897     0.709718     0.000000
N(1)       0.241239     1.324758     0.000000
H(1)       0.224165     2.335424     0.000000
C(1)       1.507368     0.726274     0.000000
O(1)       2.518005     1.411594     0.000000
O(1)      -2.043648     1.337449     0.000000
H(1)      -1.808257    -1.143873     0.000000
H(1)       0.182931    -2.420317     0.000000
H(1)       2.626386    -2.532891     0.000000
H(1)       3.370859    -1.183830    -0.874506
H(1)       3.370859    -1.183830     0.874506
O(2)      -3.661424    -0.883408     0.000000
H(2)      -3.462053     0.068032     0.000000
H(2)      -4.615649    -0.964529     0.000000
*
```

Here the example is a mono-hydrated thymine molecule, where the thymine is treated at the main fragment and water is treated at the environment. It will result in the following output

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1:  0.322826 au     8.785 eV   70852.1 cm**-1
  Amplitude    Excitation
  -0.689911    37 -> x
Percentage singles character=     96.89

IROOT=  2:  0.364959 au     9.931 eV   80099.2 cm**-1
  Amplitude    Excitation
  -0.689956    35 -> x
Percentage singles character=     95.49

IROOT=  3:  0.378175 au    10.291 eV   82999.9 cm**-1
  Amplitude    Excitation
  -0.691827    36 -> x
Percentage singles character=     93.93

IROOT=  4:  0.403845 au    10.989 eV   88633.7 cm**-1
  Amplitude    Excitation
  -0.690254    34 -> x
Percentage singles character=     96.55
```

The result of a full a IP-EOM-DLPNO-CCSD calculation with NORMALPNO setting would have looked like

```
IROOT=  1:  0.322576 au     8.778 eV   70797.3 cm**-1
  Amplitude    Excitation
   0.689734    37 -> x
Percentage singles character=     96.88

IROOT=  2:  0.364691 au     9.924 eV   80040.3 cm**-1
  Amplitude    Excitation
  -0.689947    35 -> x
Percentage singles character=     95.50

IROOT=  3:  0.377966 au    10.285 eV   82954.0 cm**-1
  Amplitude    Excitation
  -0.691801    36 -> x
Percentage singles character=     93.94

IROOT=  4:  0.402497 au    10.953 eV   88337.9 cm**-1
  Amplitude    Excitation
  -0.690138    34 -> x
Percentage singles character=     96.50
```

The results in multi-layer IP-EOM-DLPNO-CCSD method has been found to be in excellent agreement with standard variant. The $MP2FragInter$ treatment is not available for the EOM method. To get a reasonable accuracy one need to treat the fragment from where the ionization is happening (thymine in the above example) at the highest possible level. The interaction between the main fragment (thymine) and environment (water) should be treated at the intermediate level accuracy. The environment can safely be treated with $HFFragInter$ for almost all the cases. One can decide the size of the main fragment by looking at HF occupied orbitals as the Koopmans' approximation is a very good zeroth order guess for the IP values. The electron attached states are much less localized as compared to the ionization problem. Consequently, the multi-layer EA-EOM-DLPNO-CCSD requires much more tighter thresholds than the IP variant. An typical input file for multi-layer EA-EOM-DLPNO-CCSD will look as follows.

```
! EA-EOM-DLPNO-CCSD NORMALPNO  ma-def2-SVP RIJCOSX aug-cc-pVDZ/C def2/J

%mdci
  NRoots             4
  FollowCIS          true
```

(continues on next page)

(continued from previous page)

```
  TCutPNOSingles     1e-12
  MaxIter            2000
  DTol               1e-7
  NDAV               10
  NormalPNOFragInter { 1 1 }
  LoosePNOFragInter  { 1 2 } { 2 2 }
end

* xyz 0 1
N(1)   -1.114   -0.934   -3.554
C(1)   -0.343   -0.202   -4.483
H(1)   -0.668   -0.311   -5.520
C(1)   0.635   1.107   -2.633
O(1)   1.241   2.018   -2.013
N(1)   0.022   0.050   -1.776
H(1)   -0.069   0.339   -0.800
C(1)   -0.975   -0.782   -2.233
O(1)   -1.697   -1.422   -1.333
C(1)   1.087   1.852   -4.986
H(1)   1.673   2.594   -4.418
H(1)   1.771   1.340   -5.697
H(1)   0.348   2.403   -5.609
H(1)   -1.823   -1.635   -3.888
N(2)   -4.904   -4.773   -4.958
H(2)   -4.634   -3.987   -5.572
C(2)   -4.875   -4.201   -3.599
C(2)   -3.704   -3.226   -3.310
H(2)   -5.818   -3.646   -3.423
H(2)   -4.859   -5.012   -2.850
O(2)   -3.026   -2.826   -4.301
H(2)   -4.078   -5.386   -5.029
O(2)   -3.559   -2.899   -2.077
H(2)   -2.494   -2.057   -1.754
C(2)   0.440   0.898   -4.018
end
```

It is a thymine-glycine complex where the thymine is treated as the main fragment and glycine as the environment. One needs to use a more tighter value of $TCutPNOSingles$ for EA as in the case of standard EA-EOM-DLPNO-CCSD. The $TCutPNOSingles$ for the respective fragments automatically gets adjusted based on their respective $TCutPNO$ values. The output will be

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: -0.039822 au    -1.084 eV   -8739.9 cm**-1
  Amplitude    Excitation
   0.689012     x ->  53
Percentage singles character=     93.33

IROOT=  2:  0.025156 au     0.685 eV    5521.0 cm**-1
  Amplitude    Excitation
  -0.614385     x ->  54
   0.139410     x ->  55
   0.297903     x ->  59
Percentage singles character=     96.86

IROOT=  3:  0.044569 au     1.213 eV    9781.7 cm**-1
  Amplitude    Excitation
   0.116867     x ->  54
   0.684240     x ->  55
```

(continues on next page)

(continued from previous page)

```
Percentage singles character=     98.16

IROOT=  4:  0.053677 au     1.461 eV   11780.7 cm**-1
  Amplitude    Excitation
   0.695211     x ->  56
Percentage singles character=     98.12
```

The results are in excellent agreement with the standard EA-EOM-DLPNO-CCSD method.

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: -0.038862 au    -1.057 eV   -8529.3 cm**-1
  Amplitude    Excitation
  -0.689412     x ->  53
Percentage singles character=     93.47

IROOT=  2:  0.025448 au     0.692 eV    5585.2 cm**-1
  Amplitude    Excitation
  -0.654101     x ->  54
   0.131404     x ->  55
   0.207630     x ->  59
Percentage singles character=     97.47

IROOT=  3:  0.044651 au     1.215 eV    9799.8 cm**-1
  Amplitude    Excitation
   0.112183     x ->  54
   0.684562     x ->  55
Percentage singles character=     98.17

IROOT=  4:  0.053780 au     1.463 eV   11803.3 cm**-1
  Amplitude    Excitation
   0.695635     x ->  56
Percentage singles character=     98.16
```

To get the reasonable accuracy in multi-layer EA-EOM-CCSD one need to treat the environment and inter-fragment interaction atleast at $LoosePNOFragInter$ level.

3.10.13 Excited States (EOM/STEOM/ADC)

The computation of excited states is described in section *Excited States Calculations*.

3.10.14 Keywords

The keywords for the "simple" input lines are listed in Table 3.43. The `%mdci` block options are listed below:

```
%mdci
  citype CISD     # CI singles+doubles
    QCISD    # quadratic CI (singles+doubles)
    CCSD     # coupled-cluster singles+doubles
    CEPA_1   # coupled-electron pair approximation ''1''
    CEPA_2   #
    CEPA_3   #
    NCEPA_1  # our slightly modified versions of CEPA
    NCEPA_2  # and CPF
    NCEPA_3  #
    NCPF_1   #
```

(continues on next page)

(continued from previous page)

```
 NCPF_2   #
 NCPF_3   #
 ACPF     # averaged coupled-pair functional
 ACPF_2   # Gdanitz modification of it
 NACPF    # our modification of it
 AQCC     # Szalay + Bartlett
 SEOI     # a strictly size extensive energy functional
          # maintaining unitary invariance (not yet
          # available for UHF)

ewin -3,1e3   # orbital energy window to determine which
              # MOs are included in the treatment
              # (respects settings in %method block)
Singles true  # include single excitations in the
              # treatment (default true)
Triples 0     # (T) correction in CCSD(T)/QCISD(T)
              # default is no triples
        1     # Algorithm 1 (lots of memory, fast)
        2     # Algorithm 2 (less memory, about 2x slower,
              # not yet available for UHF)
Brueckner true   # use Brueckner orbitals
                 # (default false)
Denmat none       # no evaluation of density matrices
       linearized # density matrix obtained by retaining
                  # only CEPA_0-like terms, i.e., those
                  # linear in the excitation amplitudes
       unrelaxed  # unrelaxed density matrices, i.e.,
                  # density matrices without orbital
                  # relaxation
       orbopt     # perform orbital optimization yielding
                  # fully relaxed density matrices (if
                  # citype chosen as CCSD or QCISD this option
                  # implies evaluation of the Z vector).
                  # (default: linearized)
ZSimple true     # simplified evaluation of the Z vector
                 # in case of orbital optimized CCD
                 # (citype chosen as CCSD or QCISD and
                 # Denmat as orbopt) by using an
                 # analytical formula
        false    # explicit solution of Z vector
                 # equations
                 # in case of orbital optimized CCD
                 # (default: false)
UseQROs          # use of quasi-restricted orbitals
                 # (default false)
Localize 0    # use localized MOs. Presently very little
              # use is made of locality. It may help
              # for interpretations. Localization is
              # incompatible with the (T) correction
         PM   # Use Pipek-Mezey localized MOs
         FB   # use Foster-Boys localized MOs
NatOrbIters 0 # Perform natural orbital iterations.
              # default is none. Not possible for CCSD
              # and QCISD
pCCSDAB       # the three parameters for parametrized
pCCSDCD       # coupled-cluster (default is 1.0 which
pCCSDEF       # corresponds to normal CCSD
# this defines how the rate limiting step is handled
# MO and AOX need lots of disk and I/O but if they
# can be done they are fast
KCOpt KC_MO    # Perform full 4-index transformation
```

(continues on next page)

(continued from previous page)

```
        KC_AOBLAS# AO direct with BLAS (preferred)
                 # (not yet available for UHF, switch to KC_AOX)
        KC_AO    # AO direct handling of 3,4 externals
                 # (not yet available for UHF, switch to KC_AOX)
        KC_RI    # RI approximation of 3,4 externals
                 # (not yet available for UHF)
        KC_RI2   # Alternative RI (not recommended)
                 # (not yet available for UHF)
        KC_AOX   # Do it from stored AO exchange integrals
   PrintLevel 2  # Control the amount of output. For 3 and
                 # higher things like pair correlation
                 # energies are printed.
   MaxIter 35    # Max. number of iterations
   # How the integral transformation is done.
   # Note that it is fine to do AOX or AO or AOBLAS
   # together with trafo_ri
   TrafoType trafo_jk # Partial trafo to J+K operators
             trafo_ri   # RI transformation of all
                        # integrals up to 2-externals
                        # (3-ext for (T))and rest on the
                        # fly
             trafo_full # Full four index transformation.
                        # Automatically chosen for
                        # KCOpt=KC_MO
   MaxCore 350 # Memory in MB - used for integral
                # trafos and batching and for storage of
                # integrals and amplitudes
                # don't be too generous
   STol  1e-5   # Max. element of the residual vector
                # for convergence check
   LShift 0.3   # Level shift to be used in update of
                # coefficients
   MaxDIIS  7   # Max number of DIIS vectors to be stored
   # this lets you control how much and what is residing
   # in central memory. May speed up things. Note that
   # MaxCore is not respected here
   InCore   0   # nothing in core
            1   #  + sigma-vector and amplitudes (default)
            2   #  + Jij(a,b) Kij(a,b) operators
            3   #  + DIIS vectors
            4   #  + 3-exernal integral Kia(b,c)
            5   #  + 4-external integrals Kab(c,d)
                # this is identical to ALL
            # the default is AUTO which means that incore
            # is chosen based on MaxCore

    #--------------------------------
    # DLPNO specific settings:
    # See local correlation section!
    #--------------------------------

end
```

Table 3.43: Simple input keywords for the MDCI module.

Keyword	Description
Canonical method selection	
`CCSD`	Coupled-cluster singles and doubles
`CCSD(T)`	Same with perturbative triples correction
`CCSD-F12`	CCSD with F12 correction
`CCSD(T)-F12`	CCSD(T) with F12 correction
`CCSD-F12/RI`	CCSD with RI-F12 correction

continues on next p

Table 3.43 – continued from previous page

Keyword	Description
CCSD-F12D/RI	CCSD with RI-F12 correction employing the D approximation (less expensive)
CCSD(T)-F12/RI	CCSD(T) with RI-F12 correction
CCSD(T)-F12D/RI	CCSD(T) with RI-F12 correction employing the D approximation (less expensive)
QCISD	Quadratic Configuration interaction
QCISD(T)	Same with perturbative triples correction
QCISD-F12	QCISD with F12 correction
QCISD(T)-F12	QCISD(T) with F12 correction
QCISD-F12/RI	QCISD with RI-F12 correction
QCISD(T)-F12/RI	QCISD(T) with RI-F12 correction
NCPF/1	A "new" modified coupled-pair functional
CEPA/1	Coupled-electron-pair approximation
NCEPA/1	The CEPA analogue of NCPF/1
RI-CEPA/1-F12	RI-CEPA with F12 correction
MP3	MP3 energies
SCS-MP3	Grimme's refined version of MP3
Integral handling for the canonical methods	
AOX-<Method>	Compute the 3- and 4-external integrals from pre-stored AO integrals
AO-<Method>	Compute the 3- and 4-external contributions on the fly
MO-<Method>	(Default) Perform a full four index integral transformation
RI-<Method>	Use RI-approximation for all integrals
RI34-<Method>	Use RI-approximation for 3- and 4-external integrals
Local correlation method selection	
DLPNO-CCSD	Domain based local pair natural orbital coupled-cluster method with single and double excitat
DLPNO-CCSD(T)	DLPNO-CCSD with perturbative triple excitations
DLPNO-CCSD(T1)	DLPNO-CCSD with iterative perturbative triple excitations
DLPNO-CCSD-F12	DLPNO-CCSD with F12 correction employing an efficient form of the C approximation
DLPNO-CCSD-F12/D	DLPNO-CCSD-F12 with approach D (less expensive than the C approximation)
DLPNO-CCSD(T)-F12	DLPNO-CCSD(T) with F12 correction employing an efficient form of the C approximation
DLPNO-CCSD(T)-F12/D	DLPNO-CCSD(T)-F12 with approach D (less expensive than the C approximation)
DLPNO-CCSD(T1)-F12	DLPNO-CCSD(T1) with F12 correction employing an efficient form of the C approximation
DLPNO-CCSD(T1)-F12/D	DLPNO-CCSD(T1)-F12 with approach D (less expensive than the C approximation)
Local correlation accuracy control	
LoosePNO	Selects loose DLPNO thresholds
NormalPNO	Selects default DLPNO thresholds
TightPNO	Selects tight DLPNO thresholds
DLPNO-HFC1	Tightened truncation setting for DLPNO-CCSD hyperfine coupling constants calculation
DLPNO-HFC2	Tighter truncation setting than for DLPNO-HFC1

3.11 Correlated Methods using Automatic Code Generation (AUTOCI)

The AUTOCI module hosts a number of methods, where their manual implementation would be tedious or practically impossible. The module works with all types of reference wave function available in ORCA, i.e., RHF, ROHF, UHF and *CASSCF* and offers CI and related methods.

All the methods are implemented in canonical orbital basis and storing all integrals on disk.

3.11.1 Introduction

All the theories are obtained by the means of automated programming within the ORCA-AGE (Automated Generator Environment for ORCA).[361, 362] The CI module reads in the SCF wavefunction and optimizes the coefficient of the CI expansion. Conceptually, the module is similar to orca_mdci, therefore the input and output do have a lot in common.

3.11.2 Basis Usage / List of Features

The following single-reference methods are currently implemented in the AUTOCI.

Reference	Correlation	Energy	Gradient
RHF	CID	✓	✓
RHF	CISD	✓	✓
RHF	CISDT	✓	
RHF	CCD	✓	✓
RHF	CCSD	✓	✓
RHF	CCSD[T]	✓	✓
RHF	CCSD(T)	✓	✓
RHF	CCSDT	✓	
RHF	CEPA(0)	✓	✓
RHF	CC2	✓	
RHF	QCISD	✓	
RHF	MP2	✓	✓
RHF	MP3	✓	✓
RHF	MP4	✓	✓
RHF	MP4(SDQ)	✓	✓
UHF	CID	✓	✓
UHF	CISD	✓	✓
UHF	CISDT	✓	✓
UHF	CCD	✓	✓
UHF	CCSD	✓	✓
UHF	CCSD[T]	✓	✓
UHF	CCSD(T)	✓	✓
UHF	CCSDT	✓	✓
UHF	CCSDTQ	✓	
UHF	CCSDT-1	✓	
UHF	CCSDT-2	✓	
UHF	CCSDT-3	✓	
UHF	CCSDT-4	✓	
UHF	CEPA(0)	✓	✓
UHF	CC2	✓	
UHF	CC3	✓	
UHF	QCISD	✓	
UHF	MP2	✓	✓
UHF	MP3	✓	✓
UHF	MP4	✓	✓
UHF	MP4(SDQ)	✓	✓
UHF	MP5	✓	✓
ROHF	CISD	✓	

Any AUTOCI single-reference methods can be called from the simple keyword by prepending `AUTOCI-` to the correlation method, for instance

```
! AUTOCI-CCSD
```

The prefix 'AUTOCI' is added to dinstinguish from the handwritten implementations in `orca_mdci`and `orca_mp2`.

Warning

Higher order CC calculations like CCSDT and CCSDTQ have extreme memory requirements in the CC Iterations. For UHF CCSDTQ particularly (computational efford scales as N^{10} !) only systems with few electrons and basis

functions are feasible depending on your achitecture.

For a CASSCF reference wavefunction, the prefix is omitted. The following multi-reference methods and their unrelaxed densities are available.

Reference	Correlation	Energy	Unrelaxed Density
CASSCF	FIC-MRCI	✓	✓
CASSCF	FIC-MRCEPA(0)	✓	✓
CASSCF	FIC-MRACPF	✓	✓
CASSCF	FIC-MRAQCC	✓	✓
CASSCF	FIC-MRCC	✓	Linearized
CASSCF	FIC-DDCI3	✓	✓
CASSCF	FIC-NEVPT2	✓	✓
CASSCF	FIC-NEVPT3	✓	✓
CASSCF	FIC-NEVPT4(SD)	✓	✓

Note

If a multi-reference method is specified as a simple keyword, all information about reference spaces, number of roots etc. is taken from the CASSCF module that is assumed to be run in advance. More refined settings require the `%autoci` block in the input.

3.11.3 Analytic Nuclear Gradients with AUTOCI

Obtaining accurate geometries is crucial to computing molecular properties accurately. In order to perform geometry optimisations, the nuclear gradient is necessary and while this can easily be obtained using numerical finite difference methods, it is also quite costly. More importantly, perhaps, is the fact that numeric derivatives tend to become unstable. Therefore, being able to evaluate analytic gradients is of vital importance. Using the AGE, a general framework has been built that supports arbitrary-order CI, CC and MPn nuclear gradients (and other derivatives).[362]

An example is shown below how to optimise a geometry using AUTOCI's gradients at the CCSD level of theory

```
! RHF cc-pVTZ AUTOCI-CCSD VerytightSCF Opt

%maxcore 10000

*xyz 0 1
...
*
```

The analytic gradients can even be used to perform semi-numerical frequency calculations

```
! AUTOCI-CCSD NumFreq
```

Besides nuclear gradients, all other first-order properties available in ORCA are available for the respective methods, such as dipole/quadrupole moments, hyperfine couplings or quadrupole splittings. As discussed above, (un)relaxed densities can be requested via

```
%autoci
  density relaxed
end
```

For geometry optimisations, both the unrelaxed and relaxed densities are computed automatically and do not need to be requested explicitly.

3.11.4 AUTOCI Response Properties via Analytic Derivatives

For single-reference methods (`CCSD` and `CCSD(T)`), some response properties could be calculated via the analytic derivative scheme.

Response Property	Reference	Correlation	Orbital Response
Static Polarizabilities*	RHF	CCSD	Unrelaxed / Relaxed
	RHF	CCSD(T)	Unrelaxed / Relaxed
	UHF	CCSD	Unrelaxed / Relaxed
	UHF	CCSD(T)	Unrelaxed / Relaxed
NMR Shielding (GIAO)	RHF	CCSD	Unrelaxed / Relaxed
	RHF	CCSD(T)	Unrelaxed / Relaxed
	UHF	CCSD	Unrelaxed / Relaxed
	UHF	CCSD(T)	Unrelaxed / Relaxed
G-Tensor (GIAO)	UHF	CCSD	Unrelaxed / Relaxed
	UHF	CCSD(T)	Unrelaxed / Relaxed

(* dipole-dipole, dipole-quadrupole, quadrupole-quadrupole) Note that for NMR shielding and EPR g-Tensor the fully analytic GIAO approach (see *Nuclear Magnetic Resonance (NMR) Parameters* and *Gauge origin treatment*) is the default and the common gauge approach (less accurate) is not available.

To calculate these properties, the following quantities need to be computed:

- Unperturbed CC amplitudes
- Unperturbed CC Λ (Lagrange Multiplier)
- Unperturbed 1-RDM
- Unperturbed 2-RDM (†)
- Unperturbed Z-Vector (†)
- Perturbed CC amplitudes
- Perturbed CC Λ (Lagrange Multiplier)
- Perturbed 1-RDM
- Perturbed 2-RDM (†)
- Perturbed Z-Vector (†)

Tasks labelled with (†) are optional - they are only required for the "relaxed" scheme of computing response properties. With the "relaxed" scheme, the orbitals are allowed to relax to external perturbations. (See: *orbital relaxation*

Unperturbed Parameters

From CC gradient theory and as stated in the previous *MDCI section*, the CC energy Lagrangian could be written as (denoting the similarity-transformed Hamiltonian as $\bar{H} = e^{-T} H e^{T}$):

$$L_{\text{CC}} = \langle \Phi_0 | \bar{H} | \Phi_0 \rangle + \sum_{\eta} \lambda_{\eta} \langle \Phi_{\eta} | \bar{H} | \Phi_0 \rangle + \sum_{ai} f_{ai} z_{ai}$$

Equivalently, we could also express this Lagrangian in terms of density matrices:

$$\begin{aligned} L_{\text{CC}} &= \sum_{pq} h_{pq} \frac{\partial L_{\text{CC}}}{\partial h_{pq}} \\ &+ \sum_{pqrs} (pq|rs) \frac{\partial L_{\text{CC}}}{\partial (pq|rs)} \\ &= \sum_{pq} h_{pq} D_{pq} + \sum_{pqrs} (pq|rs) d_{pqrs} \end{aligned}$$

in which the one- and two-body reduced density matrices (RDMs) could be decomposed into the amplitude and orbital contributions:

$$\begin{aligned} D_{pq} &= \gamma_{pq} + z_{pq} \delta_{pa} \delta_{qi} \\ d_{pqrs} &= \Gamma^{pr}_{qs} + z_{pq} \sum_{k} \left(\delta_{pa} \delta_{qi} \delta_{rk} \delta_{sk} - \delta_{pa} \delta_{qk} \delta_{rk} \delta_{si} \right) \end{aligned}$$

The above are known as the relaxed 1- and 2-RDMs, since the terms involving the z_{pq} arise due to *orbital relaxation*. The CC amplitude contributions (a.k.a. the unrelaxed 1- and 2-RDMs) are defined as:

$$\begin{aligned} \gamma_{pq} &= \langle \Phi_0 | (1 + \Lambda) e^{-T} E^p_q e^{T} | \Phi_0 \rangle \\ \Gamma^{pr}_{qs} &= \langle \Phi_0 | (1 + \Lambda) e^{-T} \left(E^p_q E^r_s - E^p_s \delta_{rq} \right) e^{T} | \Phi_0 \rangle \end{aligned}$$

These quantities depend on the CC amplitudes t_{η} and Lagrange multipliers λ_{η}, for which we need to solve the amplitude- and Λ-equations:

$$\begin{aligned} \langle \Phi_{\eta} | \bar{H} | \Phi_0 \rangle &= 0 \\ \langle \Phi_0 | (1 + \Lambda) (\bar{H} - E_{\text{CC}}) | \Phi_{\eta} \rangle &= 0 \end{aligned}$$

Perturbed Parameters

The well-known Wigner's $2n + 1$ rule of perturbation theory states: wavefunction parameters up to the n-th order are required for the energies at the order of $2n + 1$. For non-variational methods like CC, the Lagrangian approach is employed, resulting in a variational energy functional (albeit non-Rayleigh-Ritz, thus no guaranteed upper bound), which enables the analytic evaluation of derivatives. The $2n + 2$ rule of the Lagrange multiplier in such Lagrangian functional states: Lagrange multipliers up to the n-th order are required for the energies at the order of $2n + 2$.

In practice, and here in `ORCA-AUTOCI` implementation, these two rules (especially the $2n + 2$ rule) are not fully exploited. This is due to the consideration of computational efficiency for certain molecular propeties. For example, the general expression for the second derivative of the CC energy, subject to two external perturbations x and y, could be written as:

$$\begin{aligned} \frac{\mathrm{d}^2 E}{\mathrm{d}x \mathrm{d}y} &= \langle \Phi_0 | (1 + \Lambda) e^{-T} \frac{\partial^2 H}{\partial x \partial y} e^{T} | \Phi_0 \rangle \\ &+ \langle \Phi_0 | (1 + \Lambda) \left[e^{-T} \frac{\partial H}{\partial x} e^{T}, \frac{\mathrm{d}T}{\mathrm{d}y} \right] | \Phi_0 \rangle \\ &+ \langle \Phi_0 | (1 + \Lambda) \left[e^{-T} \frac{\partial H}{\partial y} e^{T}, \frac{\mathrm{d}T}{\mathrm{d}x} \right] | \Phi_0 \rangle \\ &+ \langle \Phi_0 | (1 + \Lambda) \left[\left[e^{-T} H e^{T}, \frac{\mathrm{d}T}{\mathrm{d}x} \right], \frac{\mathrm{d}T}{\mathrm{d}y} \right] | \Phi_0 \rangle \end{aligned}$$

which is known as the "symmetric" expression, satisfying both the $2n+1$ and $2n+2$ rules. However, the following "asymmetric" expression might be preferred for mixed derivatives:

$$\begin{aligned}
&\frac{\mathrm{d}^2 E}{\mathrm{d}x\mathrm{d}y} \\
&= \langle\Phi_0|(1+\Lambda)e^{-T}\frac{\partial^2 H}{\partial x\partial y}e^{T}|\Phi_0\rangle \\
&+\langle\Phi_0|(1+\Lambda)\left[e^{-T}\frac{\partial H}{\partial x}e^{T},\frac{\mathrm{d}T}{\mathrm{d}y}\right]|\Phi_0\rangle \\
&+\langle\Phi_0|\frac{\mathrm{d}\Lambda}{\mathrm{d}y}e^{-T}\frac{\partial H}{\partial x}e^{T}|\Phi_0\rangle
\end{aligned}$$

In this way, we solve the perturbed T and Λ equations subject to only a single perturbation. For NMR shielding this means we can solve these equations only for the magnetic-field perturbation and there are 3 equations for $\mathrm{d}T/\mathrm{d}B_i$ and 3 for $\mathrm{d}\Lambda/\mathrm{d}B_i$. If we were to use the first "symmetric" expression, although we could save some effort by not having to solve for any perturbed Λ, we still need to solve for perturbed T, subject to both the magnetic-field perturbation (3 equations) and the nuclear magnetic moment perturbation ($3N_{\text{atoms}}$ equations).

To obtain the equations for perturbed T and Λ, we differentiate the unperturbed equations, w.r.t. external perturbation χ

$$\begin{aligned}
\frac{\partial}{\partial\chi}\langle\Phi_\eta|\bar{H}|\Phi_0\rangle &= 0 \\
\frac{\partial}{\partial\chi}\langle\Phi_0|(1+\Lambda)(\bar{H}-E_{\text{CC}})|\Phi_\eta\rangle &= 0
\end{aligned}$$

Then we build the perturbed 1- and 2-RDMs, by taking derivative of the unperturbed expressions:

$$\begin{aligned}
\frac{\partial\gamma_{pq}}{\partial\chi} &= \frac{\partial}{\partial\chi}\langle\Phi_0|(1+\Lambda)e^{-T}E^p_q e^{T}|\Phi_0\rangle \\
\frac{\partial\Gamma^{pr}_{qs}}{\partial\chi} &= \frac{\partial}{\partial\chi}\langle\Phi_0|(1+\Lambda)e^{-T}\left(E^p_q E^r_s - E^p_s\delta_{rq}\right)e^{T}|\Phi_0\rangle
\end{aligned}$$

The expressions and relevant modules are produced by the code generator. Note that for the currently available response properties - polarizabilities, NMR shielding, EPR g-tensors - which are technically one-particle properties, for which the 2-RDMs do not appear explicitly in the energy derivative expression, both the unperturbed 2-RDMs and the perturbed 2-RDMs are only needed when *orbital relaxation* needs to be computed.

Orbital Relaxation

By including the orbital relaxation effect into property calculations, we essentially allow the MO to respond to external perturbations. This leads to the following parameterization of MO coeffcients:

$$\mathbf{C}(\chi) = \mathbf{C}(0)\mathbf{U}(\chi)$$

then MO derivatives could be written as:

$$\mathbf{C}^{\chi} \equiv \frac{\partial\mathbf{C}(\chi)}{\partial\chi} = \mathbf{C}(0)\frac{\partial\mathbf{U}(\chi)}{\partial\chi} \equiv \mathbf{C}\mathbf{U}^{\chi}$$

where $\mathbf{U}^{\chi}$ are solutions to the *CPSCF* equations. For unrelaxed calculations the MO coefficients are always $\mathbf{C}(0)$. In relaxed calculations, the MO response enters relevant CC equations in two ways.

First, as described *above*, for second-order response properties (like polarizabilities, NMR shielding, etc.) we need to solve the first-order T and Λ equations. Within a relaxed scheme, any derivative integrals (Fock, ERI, etc.) entering

the CC equations contain CPSCF contributions. For example:

$$\frac{\partial(pq|rs)}{\partial\chi} = \sum_{\mu\nu\kappa\tau} C_{\mu p}C_{\nu q}\frac{\partial(\mu\nu|\kappa\tau)}{\partial\chi}C_{\kappa r}C_{\tau s} + \sum_{\mu\nu\kappa\tau} C^{\chi}_{\mu p}C_{\nu q}(\mu\nu|\kappa\tau)C_{\kappa r}C_{\tau s} + \sum_{\mu\nu\kappa\tau} C_{\mu p}C^{\chi}_{\nu q}(\mu\nu|\kappa\tau)C_{\kappa r}C_{\tau s} + \sum_{\mu\nu\kappa\tau} C_{\mu p}C_{\nu q}(\mu\nu|\kappa\tau)C^{\chi}_{\kappa r}C_{\tau s} + \sum_{\mu\nu\kappa\tau} C_{\mu p}C_{\nu q}(\mu\nu|\kappa\tau)C_{\kappa r}C^{\chi}_{\tau s}$$

For unrelaxed calculations the last 4 terms are zero.

Second, to compute the relaxed 1- and 2-RDMs and their derivatives, the z-vector contribution needs to be included. This is a consequence of requiring the CC energy to be invariant to orbital rotation, in the presence of external perturbations. This nice trick by Schaefer and Handy allows gradient to be evaluated without having to solve $3N_{atoms}$ equations for the orbitals. Instead, only the so-called "z-vector" equation needs to be solved. It thus opens a door towards efficient analytic response property evaluations. The z-vector equations are of the same form as the *CPSCF* equations:

$$\mathbf{Az} = \mathbf{X}$$

where the **A** matrix is exactly the same as the CPSCF LHS, and the RHS matrix **X** is the orbital gradient of the CC energy functional. For second-order response properties, the derivatives (subject to each perturbation requested) of this equation also need to be solved.

The discussion of relaxed properties vs unrelaxed ones are quite common. Generally, we believe that for electric-field dependent properties like polarizabilities, inclusion of orbital relaxation does not necessarily lead to more accurate values compared with experiments. However it is generally regarded essential to include orbital relaxation for magnetic-field dependent properties like NMR and EPR parameters.

Finally, it is also worth noting that this orbital relaxtion scheme does not recover the MO response of using HF orbitals in a CC wavefunction setup, which relies on the orbital-optimized coupled-cluster (OOCC) methods, as described in the *previous section*.

Input Options

The input parameters for the wavefunctions are controlled by the `%autoci` block, while the property-related parameters are controlled by respective property blocks, like `elprop` and `eprnmr`. Some useful options are gathered below.

```
%autoci
    citype
        CCSD(T)
        CCSD
    STol      1e-06 # residue convergence tolerance
    MaxIter   50    # maximum number of iterations
    MaxDIIS   5     # depth of the DIIS memory
    density         # need at least unrelaxed density (see details below)
        unrelaxed
        relaxed
end

%elprop
```

(continues on next page)

(continued from previous page)

```
    polar         true    # dipole-dipole polarizability
    polardipquad  true    # dipole-quadrupole polarizability
    polarquadquad true    # quadrupole-quadrupole polarizability
end

%eprnmr
    gtensor       true    # calculate the g-tensor
    NMRShielding  1       # for chosen nuclei - specified with the Nuclei keyword
                  2       # for all nuclei, equivalent to the 'NMR' simple input
end
```

Output files: Calculation outputs generated from AutoCI-Response modules are re-directed into separate outputfiles, with the names `basename.autociresp_efield`, `basename.autociresp_bfield`, etc.

Input Examples

Highly accurate NMR shielding for HF:

```
! AUTOCI-CCSD(T) NMR pcsseg-4 cc-pwcv5z/c ExtremeSCF

%autoci
  STol 1e-10
  density relaxed
end

%eprnmr
  Tol 1e-12
end

%coords
  CTyp xyz
  Charge 0
  Mult 1
  Units bohrs
  coords
    H  0.00000000  0.00000000  1.64411926
    F  0.00000000  0.00000000 -0.08721704
  end
end
```

G-Tensor for NH radical, with the SOC operator being the effective nuclear charge (see *SOCoptions*):

```
! UHF cc-pVTZ G-Tensor ExtremeSCF AUTOCI-CCSD(T)
! NoFrozenCore Bohrs

%rel
  SOCType 1
  Zeff[1] = 1.0   # H
  Zeff[7] = 4.55  # N
end

%eprnmr
  Tol 1e-10
end

%autoci
  STol 1e-9
  Density relaxed
end
```

(continues on next page)

(continued from previous page)

```
# NH radical
* xyz 0 3
N 0.0  0.0  0.0
H 0.0  0.0  1.957794
*
```

By changing the `%rel` block to the following we can do the same calculation, using a more accurate SOC description (SOMF):

```
%rel
  # these two are defaults
  SOCType 3
  SOCFlags 1,4,3,0
end
```

3.11.5 Fully Internally Contracted MRCI (FIC-MRCI)

Starting point for any multireference approach is a reference wavefunction that consists of multiple determinants or configurations state functions (CSFs). In many instances this is the complete active space SCF (CASSCF) wavefunction. In the uncontracted MRCI approach, as implemented in the `orca_mrci` module, the wavefunction is expanded in terms of excited CSFs that are generated by considering excitations with respect to all reference CSFs. The methodology scales with the number of reference CSFs and hence is restricted to small reference spaces. Moreover, the configuration driven algorithm used in `orca_mrci` keeps all integrals in memory, which further limits the overall size of the molecule.

Internal contraction as proposed by Meyer and Siegbahn avoids these bottlenecks [418, 419]. Here, excited CSFs are generated by applying the excitation operator to the reference wavefunction as whole. The fully internally contracted MRCI (FIC-MRCI) uses the same internal contraction scheme as the FIC-NEVPT2 (aka PC-NEVPT2). The entire methodology as well as a comparison with the conventional uncontracted MRCI is reported in our article [420]. The CEPA0, ACPF and AQCC variants are straight forward adoptions [421]. The residue of the FIC-MRCI ansatz

$$R_K = \langle \Phi_{qs}^{pr} | \hat{H} - E^{\text{CAS}} - \lambda E_c | \Psi^{\text{FIC}} \rangle \,, \tag{3.130}$$

is modified by the factor

$$\lambda = \begin{cases} 1 & \text{MRCI} \\ 0 & \text{CEPA0} \\ \frac{2}{N_e} & \text{ACPF} \\ \frac{1-(N_e-3)(N_e-2)}{N_e \cdot (N_e-1)} & \text{AQCC} \end{cases}$$

Here, E_c is the correlation energy and Φ_{qs}^{pr} denote the internally contracted CSF that arise from the action of the spin-traced excitation operators on the CAS-CI reference wave function

$$\Phi_{qs}^{pr} = E_q^p E_s^r \, |\Psi^{\text{CAS}}\rangle \,.$$

In case of ACPF and AQCC, the λ factor explicitly depends on the number of correlated electrons (N_e).

The general input structure is like that of the CASSCF module, e.g., the following example input reads an arbitrary set of orbitals and starts the FIC-MRCI calculation. The internal contracted formalism requires CAS-CI reduced densities up to fourth-order, which can be expensive to construct. By default, the density construction is sped up using the prescreening (PS) approximation reported in Section *N-Electron Valence State Perturbation Theory (NEVPT2)*.[422]

```
!def2-tzvp moread allowrhf noiter nofrozencore
%moinp "start.gbw"  # could be from CASSCF

%autoci
citype   FIC-MRCI      # Fully internally contracted MRCI (singles, doubles)
         FIC-MRCEPA(0) # CEPA0 version of FIC-MRCI
```

(continues on next page)

(continued from previous page)

```
         FIC-MRACPF    # ACPF  version of FIC-MRCI
         FIC-MRAQCC    # AQCC  version of FIC-MRCI
         FIC-DDCI3     # FIC-MRCI without the IJAB excitation

# CAS-CI reference wavefunction
nel 2
norb 2
mult 1,3
nroots 3,1

nthresh 1e-6  # removal of linear dependencies in the IC-CSFs
D3TPre        1e-14 # default density truncation of the 3-RDM
D4TPre        1e-14 # default density truncation of the 4-RDM

# Davidson correction for the FIC-MRCI
DavidsonOpt 0  # none (default)
            1  # Davidson correction
end
```

Currently, the program is capable of computing total energies and vertical excitation energies. More features will be available with future releases.

Input Example

```
# potentially "!FIC-MRCEPA(0)" omitting the %autoci block.
!def2-SVP

# get CASSCF orbitals and define states
%casscf
  nel 8
  norb 6
  mult 3
  nroots 1
end

# run FIC-MRCEPA(0)
%autoci
  # CAS settings are automatically copied from the %casscf block, if
  # not defined in %autoci

  citype FIC-MRCEPA(0)
end

# O2 molecule
*xyz 0 3
O 0 0 0
O 0 0 1.28
*
```

3.11.6 Fully Internally Contracted MRCC (FIC-MRCC)

Several approaches have been taken towards extending CC theory to work with genuinely multiconfigurational reference wave functions [423], yet none of these approaches has found widespread adoption. As of 2011, the internally contracted MRCC theory has had a revival, with a rigorous theoretical investigation of several approximations that also proved its orbital invariance [424] and a first report of a polynomial-scaling code obtained through automatic equation generation [425].

Our implementation in ORCA is akin to the previously published formulations in Refs. [424, 425], although everything is formulated rigorously in terms of the spin-free excitation operators $\hat{E}^p_q = \hat{a}^{p\alpha}\hat{a}_{q\alpha} + \hat{a}^{p\beta}\hat{a}_{q\beta}$, using an improved version of the ORCA-AGE code generator.[362] To begin with, the *ansatz* for the wave function is

$$|\Psi\rangle = \mathrm{e}^{\hat{T}}|0\rangle \ , \tag{3.131}$$

where $|0\rangle$ denotes a zeroth order CASSCF reference wave function and the cluster operator can be written as (Einstein's summation convention implied)

$$\begin{aligned}\hat{T} = \frac{1}{2}t^{ij}_{ab}\hat{E}^a_i\hat{E}^b_j + t^{it}_{ab}\hat{E}^a_i\hat{E}^b_t + \frac{1}{2}t^{tu}_{ab}\hat{E}^a_t\hat{E}^b_u + t^{ij}_{at}\hat{E}^a_i\hat{E}^t_j + t^{it}_{au}\hat{E}^a_i\hat{E}^u_t \\ + t^{it}_{ua}\hat{E}^u_i\hat{E}^a_t + t^{tu}_{av}\hat{E}^a_t\hat{E}^v_u + \frac{1}{2}t^{ij}_{tu}\hat{E}^t_i\hat{E}^u_j + t^{it}_{uv}\hat{E}^u_i\hat{E}^v_t \ .\end{aligned}$$

Note that we do not use normal order in the cluster operator.

Inserting the *ansatz* from Eq. (3.131) into the Schrödinger equation and pre-multiplying with the inverse exponential, we obtain the similarity transformed Hamiltonian and the energy expression,

$$\bar{H}|0\rangle = \mathrm{e}^{-\hat{T}}\hat{H}\mathrm{e}^{\hat{T}}|0\rangle = E|0\rangle \ .$$

In our code, the similarity-transformed Hamiltonian is truncated after the quadratic terms since that approximation has been found to only have minor impact on the accuracy of the method [424],

$$\bar{H} \approx \hat{H} + [\hat{H},\hat{T}] + \frac{1}{2}[[\hat{H},\hat{T}],\hat{T}] \ .$$

The residual conditions are subsequently obtained by projecting with contravariant excited functions $\langle\tilde{\Phi}_P|$ onto the Schrödinger equation,

$$r_P = \langle\tilde{\Phi}_P|\bar{H}|0\rangle \ .$$

For a definition of the contravariant projection functions, we refer to Ref. [420] since this FIC-MRCC implementation uses the same contravariant functions as the published FIC-MRCI implementation. Despite using contravariant projection functions, this is not sufficient to remove all linear dependencies from the set of projection functions $\{\tilde{\Phi}_P\}$, i.e. the metric matrix

$$S_{PQ} = \langle\tilde{\Phi}_P|\tilde{\Phi}_Q\rangle \neq \delta_{PQ}$$

has off-diagonal elements *within* excitation classes and between classes with the same number of inactive and virtual indices (ITAU and ITUA). Hence, the set of projection functions needs to be orthonormalized, which is achieved with Löwdin's canonic orthogonalization in the AUTOCI module.[1]

Input Example

The FIC-MRCC module can be started by specifying the `CIType` keyword in the `%autoci` block or by adding `FIC-MRCC` to the simple input line of an ORCA input file. The following example computes the singlet ground state energy of four hydrogen atoms arranged as a square with a side length of $2a_0$, which is commonly known as the H4 model [426].

[1] This is similar to scheme A from Ref. [425].

```
! cc-pVTZ Bohrs  # it is possible to add the `FIC-MRCC' keyword here
                 # and omit the %autoci block below

%maxcore 10000

# get CASSCF orbitals and define states.
%casscf
  nel 2
  norb 2
  mult 1
  nroots 1
end

# run FIC-MRCC
%autoci
  # CAS settings are automatically copied from the %casscf block, if
  # not defined in %autoci
  citype fic-mrcc

  # optional parameters
  nthresh 1e-6        # default removal of linear dependencies in the IC-CSFs
  D3TPre        1e-14 # default density truncation of the 3-RDM
  D4TPre        1e-14 # default density truncation of the 4-RDM
  D5TPre        1e-14 # default density truncation of the 5-RDM
  Density  linearized  # compute the linearized density (not default).
end

* int 0 1
  H 0 0 0 0.0  0.0 0.0
  H 1 0 0 2.0  0.0 0.0
  H 2 1 0 2.0 90.0 0.0
  H 1 2 3 2.0 90.0 0.0
*
```

In this example, ORCA will first run a state-specific CASSCF calculation, and then immediately continue with the fic-MRCC calculation on top of the CASSCF solution from the first step. It is, however, not required to always run a CASSCF calculation before the `autoci` module. Any ORCA `gbw/mp2nat/...` file is accepted through `%moinp`, although that route requires the user to specify the active space in the `autoci` block. `autoci` will then compute a CASCI solution with the provided input orbitals and use that information to drive the correlated calculations.

Please be aware that FIC-MRCC is a very extensive theory, which leads to long run times. The computational effort depends mainly on the number of orbitals, the number of total electrons and the size of the active space. On modestly modern hardware, calculations of $\sim$ 300 orbitals with a CAS(2,2) should be readily achievable. For larger active spaces, such as a CAS(6,6), calculations with a total of $\sim$ 200 orbitals will also complete within a day.

3.11.7 Fully Internally Contracted NEVPT (FIC-NEVPT4(SD)/FIC-NEVPT3/FIC-NEVPT2)

Perturbation theory is a cost-effective alternative to the previously mentioned FIC-MRCI and FIC-MRCC methods. Most often, a reasonable description of electron correlation is already achieved at the second order of the expansion. The CASSCF module features an efficient implementation of the second-order methods, that are able to tackle large molecules and active spaces. For more details and references on the second-order variants, we refer to the dedicated section *N-Electron Valence State Perturbation Theory (NEVPT2)*.
In the following section, the focus is on higher orders of the n-electron valence state perturbation theory (NEVPT), that are available within the AUTOCI framework. Here, all integrals are computed and kept on disk, which limits the application to approximately 700 atomic orbitals (AOs). The appearance of higher-order reduced density matrices is a common feature of internally contracted methods. Like FIC-NEVPT2, the FIC-NEVPT3 and FIC-NEVPT4(SD) require at most the fourth-order reduced density matrix. Its computation can be accelerated with the prescreening method discussed in section *N-Electron Valence State Perturbation Theory (NEVPT2)*.[422, 427] In terms of computation time, the FIC-NEVPT4(SD) is as expensive as FIC-NEVPT3 or a single iteration of the FIC-MRCI method.

FIC-NEVPT3

The FIC-NEVPT3 methodlogy is desribed in detail by Angeli et al.[428] With the first order wave function $\Psi^{(1)}$, the third order energy is given by the expression

$$E^{(3)} = \langle \Psi^{(1)} | V | \Psi^{(1)} \rangle.$$

FIC-NEVPT4(SD)

The fourth-order perturbation theory requires the second wave function, which, in principle, involves triple and quadruple excitation defining the second-order interacting space. The inclusion of the latter is computationally demanding. In the following we report an approximate solution, coined FIC-NEVPT4(SD), which is restricted to the first-order interacting space - the single and double excitations with respect to the reference wave function. This method is reported and analyzed in more detail by Kempfer et al.[429] Denoting the second order wave function with $\Psi^{(2)}$, the fourth order energy is readily expressed as

$$E^{(4)} = \langle \Psi^{(1)} | V | \Psi^{(2)} \rangle - E^{(2)} \langle \Psi^{(1)} | \Psi^{(1)} \rangle.$$

The second contribution in the expression above, $E^{(2)} \langle \Psi^{(1)} | \Psi^{(1)} \rangle$, is a renormalization term. As discussed in the paper,[429] the renormalization term is dropped (default) as it destroys the size consistancy in the absence of the exact quadruple excitations.

Input Example

```
# potentially "!FIC-NEVPT4(SD)"/"!FIC-NEVPT3" instead of the %autoci block.
!def2-tzvp

# get CASSCF orbitals and define the states
%casscf
    nel 8
    norb 6
    mult 3
    nroots 1
end


# run NEVPT4(SD)
%autoci
# CAS settings are automatically copied from the %casscf block, if
# not defined in %autoci

 citype fic-nevpt4(sd)

# optional parameters
nthresh 1e-6        # default removal of linear dependencies in the IC-CSFs
D3TPre        1e-14 # default density truncation of the 3-RDM
D4TPre        1e-14 # default density truncation of the 4-RDM

Density  Unrelaxed  # compute the unrelaxed density (not default).
end

# O2 molecule
* xyz 0 3
O 0 0 0
O 0 0 1.28
*
```

3.11.8 FIC Ansatz: Unrelaxed Densities and Natural Orbitals

The FIC-NEVPT4(SD), the FIC-MRCC and the FIC-MRCI variants have a common set excitation parameters in their wavefunction. For these methods, (linearized) unrelaxed densities are available (`density unrelaxed`). With `!KeepDens` the densities are stored in the density container (**.densities** file on disk), which can be read with the *`orca_plot` program*. The name of the densities reflect its multiplicity, root count and the actual method. Below is an example snippet of the "orca_plot jobname.densities" call:

```
---------------------
List of density names
---------------------

Index:                                                  Name of Density
-----------------------------------------------------------------------
    0:                                                     jobname.scfp
    1:                                                     jobname.scfr
    2:                                          Tdens-CAS.mult.3.root.0.p
    3:                                          Tdens-CAS.mult.3.root.1.p
    4:                                          Tdens-CAS.mult.3.root.2.p
    5:                                          Tdens-CAS.mult.3.root.0.r
    6:                                          Tdens-CAS.mult.3.root.1.r
    7:                                          Tdens-CAS.mult.3.root.2.r
    8:                      jobname.mult.3.root.0.FIC-NEVPT4(SD).autoci.p
    9:                      jobname.mult.3.root.1.FIC-NEVPT4(SD).autoci.p
   10:                      jobname.mult.3.root.2.FIC-NEVPT4(SD).autoci.p
```

Here, the densities 8-10 are unrelaxed densities of the FIC-NEVPT4(SD) method for the first three triplet roots. With the unrelaxed densities, natural orbitals (`natorbs true`) can be generated. The resulting .gbw files ending with (.nat) can be used to carry out further analysis, e.g. using the MultiWFN program.

```
!FIC-NEVPT4(sd)
...
%autoci
  Density  Unrelaxed  # compute the unrelaxed density
  natorbs true        # generate natural orbitals
end
```

3.11.9 Keywords

Table 3.45: Simple input keywords for AUTOCI methods.

Keyword	Description
`AUTOCI-<method>`	Activate single-reference correlation module from AUTOCI. Disables MDCI.
`FIC-MRCI`	
`FIC-MRCEPA(0)`	
`FIC-MRACPF`	
`FIC-MRAQCC`	
`FIC-DDCI3`	
`FIC-MRCC`	
`FIC-NEVPT2`	
`FIC-NEVPT3`	
`FIC-NEVPT4(SD)`	

Table 3.46: `%autoci` block input keywords AUTOCI methods.

Keyword	Options	Description
CIType	CID	Correlation method
	CISD	
	CISDT	
	CCD	
	CCSD	
	CCSDT	
	CEPA(0)	
	QCISD	
	CC2	
	CC3	
	CCSD[T]	
	CCSD(T)	
	CCSDT-1	
	CCSDT-2	
	CCSDT-3	
	CCSDT-4	
	FIC-MRCI	
	FIC-MRCEPA(0)	
	FIC-MRACPF	
	FIC-MRAQCC	
	FIC-DDCI3	
	FIC-MRCC	
	FIC-NEVPT2	
	FIC-NEVPT3	
	FIC-NEVPT4(SD)	
	MP2	
	MP3	
	MP4(SDQ)	
	MP4	
	MP5	
STol	1e-6	Residue convergence tolerance
LevelShift	0.2	LevelShift in the DIIS update / denominators
MaxIter	50	Maximum number of iterations
MaxDiis	5	Depth of the DIIS memory
DIISStartIter	2	Apply DIIS starting at iteration 1 (counting starts with 0).
ExcludeHigherExcD]	false	Exclude triples and higher excitations from DIIS procedure
NEl	6	Number of active electrons (CAS only)
NOrb	7	Number of active orbitals (CAS only)
Mult	1	Requested multiplicity block (CAS only)
NRoots	1	Number of roots for mult block (CAS only)
Irrep	0	Requested irrep for mult block (CAS only)
NThresh	1e-6	Threshold for lin. dependencies in the IC-CSFs basis (CAS only)
D3TPre	1e-14	Density truncation in D3 (CAS only)
D4TPre	1e-14	Density truncation in D4 (CAS only)
PrintLevel	3	Amount of printing (1-7)
TrafoType	NoRI	Type of integral transformation
	RITrafo	
KeepInts	false	Keep the transformed integrals on disk
UseOldInts	false	Use the transformed integrals found on disk
RunROHFasUHF	false	Invokes AUTOCI UHF modules with orbitals from ROHF calculation
Density	none	Type of 1-body density requested
	linearized	
	unrelaxed	
	relaxed	
Density2	none	Type of 2-body density requested
	linearized	
	unrelaxed	
	relaxed	
NatOrbs	false	Generate natural orbitals (not available for all methods)

N.B. For a ROHF reference, only CISD calculations can be performed in the current version. However, it is possible to run UHF calculations with an ROHF reference by setting the `RunROHFasUHF` flag to true. Note that this only makes sense when the reference is indeed ROHF, e.g. (calculating the isotropic part of the HFC at CCSD level, running AutoCI UHF CCSD module, with orbitals obtained from the ROHF SCF calculation):

```
! ROHF def2-svp tightscf pmodel AUTOCI-CCSD
```

(continues on next page)

(continued from previous page)

```
%autoci
    RunROHFasUHF true
end

* xyz 0 2
Cu  0.0  0.0  0.0
*

%eprnmr nuclei = all Cu {aiso} end
```

If one wishes to experiment with the module itself and the reference wavefunction stays constant, it is possible to store the transformed MO integrals on disk (`keepints`) and reuse them (`useoldints`). The program checks only whether the dimension of the integrals on disk match the problem actually solved, i.e. the user is responsible for valid data.

3.12 Arbitrary Order Coupled-Cluster (MRCC interface)

Various higher order theories (CC3, CISDT, CCSDTQ, MP4) are implemented using the *automatic code-generation*. In addition, ORCA features an interface to Kallay's powerful MRCC program. This program must be obtained separately. The interface is restricted to single point energies but can be used for rigid scan calculations or numerical frequencies.

The use of the interface is simple:

```
#
# Test the MRCC code of Mihael Kallay
#
! cc-pVDZ Conv SCFConv10 UseSym

%mrcc method   "CCSDT"
      ETol     9
      end

* xyz 0 1
F 0 0 0
H 0 0 0.95
*
```

The Method string can be any of:

```
# The excitation level specification can be anything
# like SD, SDT, SDTQ, SDTQP etc.
 %mrcc method   "CCSDT"
                "CCSD(T)"
                "CCSD[T]"
                "CCSD(T)_L"  (the lambda version)
                "CC3"
                "CCSDT-1a"
                "CCSDT-1b"
                "CISDT"
```

It is not a good idea, of course, to use this code for CCSD or CCSD(T) or CISD. Its real power lies in performing the higher order calculations. Open-shell calculations can presently not be done with the interface.

3.13 Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)

3.13.1 Introduction

The complete active space self-consistent field (CASSCF) method is a special form of a multiconfigurational SCF method and can be thought of as an extension of the Hartree-Fock method. It is a very powerful method to study static correlation effects and a solid basis for MR-CI and MR-PT treatments. It can be applied to the ground state and excited states or averages thereof. The implementation in ORCA is fairly general and also allows incomplete active spaces omitting active-active rotations. Example for such *incomplete model spaces* include arbitrary references configurations, restricted active space (RAS) and occupation restricted multiple active spaces (ORMAS). In the following section, the primary focus is on the complete active spaces.

The subject is fairly complex and ultimately require a lot of insight from the user in order to be successful. Thus, in addition to the material in the manual, we have created a CASSCF tutorial, that covers many practical tips on the calculation design and usage of the program.

There are several situations where a CASSCF treatment is a good idea:

- Wave functions with significant multireference character arising from several nearly degenerate configurations (static correlation)
- Wave functions which require a multideterminantal treatment (for example multiplets of atoms, ions, transition metal complexes,)
- Situations in which bonds are broken or partially broken.
- Generation of orbitals which are a compromise between the requirements for several states.
- Starting point for multireference methods covering dynamic correlation (NEVPT2, MRCI, MREOM, …)
- Generation of genuine spin eigenfunctions for multideterminantal/multireference wave functions.

In all of these cases, the single-determinantal Hartree-Fock/DFT methods fail badly, whereas CASSCF remains a good choice. In the latter, the orbitals are divided into three-subspaces: (a) the internal (inactive) orbitals which are doubly occupied in all configuration state functions (CSFs) (b) partially occupied (active) orbitals (c) virtual (external) orbitals which are empty in all CSFs.

A fixed number of electrons is assigned to the internal subspace and the active subspace. If N electrons are "active" in M orbitals, one speaks of a CASSCF(N,M) wave functions. All spin-eigenfunctions for N electrons in M orbitals are included in the configuration interaction step (CAS-CI) and the energy is made stationary with respect to variations in the MO and the CI coefficients. Any number of roots of any number of different multiplicities can be calculated and the CASSCF energy may be optimized with respect to a user defined average of these states.

The CASSCF method has the nice advantage that it is fully variational which renders the calculation of analytical gradients relatively easy. Thus, the CASSCF method may be used for geometry optimizations and numerical frequency calculations. It should be noted that state-averaged CASSCF calculations are not fully variational in general. The orbitals must be optimized for the state of interest.

A number of properties are available in ORCA (g-tensor, ZFS splitting, CD, MCD, susceptibility, dipoles, …). The majority of CASSCF properties such as EPR parameters are computed in the framework of the *quasi-degenerate perturbation theory*. In addition, some properties such as ZFS splittings can also be computed via perturbation theory or rigorously extracted from an effective Hamiltonian. For a detailed description of the available properties and options see section Section 3.13.15. All the aforementioned properties are computed within the CASSCF module. An exception are *Mössbauer parameters*, which are computed with the usual keywords in the `%EPRNMR` block, when the density (in the density container) is available. In general, properties within the `%EPRNMR` block are derived using linear reponse theory. An extension of the latter for a CASSCF wave function is reported in section Section 5.26 for a few selected properties.

Note

The CASSCF module hosts the *CASPT2* and *NEVPT2* methods and their variants. Higher order of multi-reference theories (e.g. *FIC-NEVPT4(sd)*) can be found in the *AutoCI module* and the *MRCI module*.

3.13.2 Theory

Denoting the state with I and its spin with S, the CASSCF wave function is written as

$$\left|\Psi_I^S\right\rangle = \sum_k C_{kI} \left|\Phi_k^S\right\rangle.$$

Here, the wave function is expanded in a set of configuration state functions (CSFs), that are a linear combination of determinants adapted to the total spin S.

To define the actual set of CSFs (Φ_I^S), the molecular orbital space is devided into three user defined subspaces:

- The "inactive orbitals" are the orbitals which are doubly occupied in all CSFs.
- The "active orbitals" are the orbitals with variable occupation numbers in the CSFs.
- The "external orbitals" are empty in all CSFs.

A complete model space, CAS(N,M), includes all CSFs, that arise from a full configuration interaction (FCI), where N active electron are distributed among the M active orbitals, while the remaining electrons of the system reside in the doubly occupied inactive orbitals. **The active orbitals are central to ansatz and must be chosen by the user with care.** The list of CSFs grows extremely quickly with the number of active electrons and orbitals (basically factorially). Depending on the actual system, the limit of feasibility is roughly around $\sim$14 active orbitals or about one million CSFs in the active space. Larger active spaces are tractable with approximate FCI solver such as the *Iterative-Configuration-ExpansionCI (ICE-CI)* or the *Density Matrix Renormalization Group (DMRG) approach*.[1] In certain situation, *incomplete model spaces such as the RAS or ORMAS* are viable alternatives.

The expansion coefficients C_{kI} represent the first set of variational parameters. Each CSF is constructed from a common set of orthonormal molecular orbitals $\psi_i(\mathbf{r})$ which are in turn expanded in basis functions $\psi_i(\mathbf{r}) = \sum_\mu c_{\mu i}\phi_\mu(\mathbf{r})$. The MO coefficients $c_{\mu i}$ form the second set of variational parameters.

The energy of the CASSCF wave function is given by the Rayleigh quotient

$$E(\mathbf{c},\mathbf{C}) = \frac{\left\langle \Psi_I^S \middle| \hat{H}_{\mathrm{BO}} \middle| \Psi_I^S \right\rangle}{\left\langle \Psi_I^S \middle| \Psi_I^S \right\rangle},$$

and represents an upper bound to the true total energy. However, CASSCF calculations are *not* designed to provide values for total energy which are close to the exact energy. The purpose of a CASSCF calculation is to provide a qualitatively correct wave function, which forms a good starting point for a treatment of dynamic electron correlation.

The CASSCF method is fully variational in the sense that the energy is made stationary with respect to variations in both sets of MO and CI coefficients. At convergence, the gradient of the energy with respect to the MO and CI coefficients vanishes

$$\frac{\partial E(\mathbf{c},\mathbf{C})}{\partial c_{\mu i}} = 0,$$

$$\frac{\partial E(\mathbf{c},\mathbf{C})}{\partial C_{kI}} = 0.$$

For a complete model space, the wave function and energy are invariant with respect to unitary transformations within the three subspaces. Hence, the orbitals within the subspaces are only defined up to a unitary transformation, and the program needs to make some canonicalization choice. In ORCA, the *final orbitals* by default are:

- natural orbitals in the active space,
- orbitals which diagonalize the CASSCF Fock matrix in the internal space and
- orbitals which diagonalize the CASSCF Fock matrix in the external space.

[1] For approximate full CI approaches, CASSCF is neither invariant to active-active rotations nor exactly size-consistent.

Optimization of CASSCF wave functions

In general, except for trivial cases, CASSCF wave functions are considerably more difficult to optimize than RHF (or UHF) wave functions. The underlying reason is that variations in **c** and **C** maybe strongly coupled and the energy functional may have many local minima in (**c**,**C**) space. Consequently, the choice of starting orbitals is of really high importance and the choice which orbitals and electrons are included in the active space has decisive influence on the success of a CASSCF study.

In general, after transformation to natural orbitals, one can classify the active space orbitals by their occupation numbers which vary between 0.0 and 2.0. In general, convergence problems are almost guaranteed if orbitals with occupation numbers close to zero or close to 2.0 are included in the active space. Occupation numbers between 0.02 and 1.98 are typically very reasonable and should not lead to large convergence problems. The reason for the occurrence of convergence problems is that the energy is only very weakly dependent on rotations between internal and active orbitals if the active orbital is almost doubly occupied and similarly for the rotations between external and weakly occupied active orbitals. However, in some cases (for example in the study of potential energy surfaces) it may not be avoidable to include weakly or almost inactive orbitals in the active space and in these cases the use of the most powerful convergence aids is necessary (*vide infra*).

With the exception, of the *TRAH-CASSCF* solver,[121] the ORCA implementation follows a *two-step procedure*, where in every cycle (Macro-Iteration) the CAS-CI problem is solved in the present set of molecular orbitals. The orbital coefficients are then updated with a unitary matrix U

$$\mathbf{c}^{\text{new}} = \mathbf{c}^{\text{old}}\mathbf{U}$$

and a new cylce begins. The process is repeated until convergence is reached, i.e., the energy does not change (default `ETol 1e-8`) or the orbital gradient is near zero (default `GTol 1e-3`). In contrast to that, a *one-step ansatz* updates the orbital coefficients and CI coefficients simultaneously.

In both cases, the unitary transformation is parametrized using an antisymmetric matrix ($X_{pq} = -X_{qp}$)

$$\mathbf{U} = \exp\left(\mathbf{X}\right),$$

where X_{pq} consists of the non-redundant orbital rotations (inter-subspace).

As in the case of single-determinant wave functions (RHF, UHF, RKS, UKS) there are first-order converging methods, such as the present default, the perturbative SuperCI (`SuperCI_PT`),[430] that only require the orbital gradients

$$\frac{\partial E\left(\mathbf{X},\mathbf{C}\right)}{\partial X_{pq}}.$$

The necesarry transformed integrals $(tu|vx)$, with tuv active and x belonging to any subspace, are a very small and readily held in central storage even for larger calculations.

On the other hand, second-order CASSCF methods, compute the Hessian and require the integrals $(pq|xy)$ and $(px|qy)$ (p,q = internal, active; x,y = any orbital). This is a fairly large set of integrals and their generation is laborious in terms of CPU time and disk storage. Standard second-order methods such as the augmented Hessian (`NR`) are more limited in the size of the molecules which can be well treated. While the `NR` is highly efficient near the minimum, it is less robust when the initial orbital are too far away. The *restricted-step augmented Hessian (`TRAH`)* mitigates many of the disadvantages and offers a more robust second-order ansatz, which can also tackle larger systems.[431] Here, most of the integrals are processed using the *RI approximation* in an AO direct strategy. For more details, we refer to original publication.

State-averaging

In many situations, it is desirable to optimize the orbitals not for a single state but for the average of several states. In order to see what is done, the energy for state I is re-written as:

$$E_I\left(\mathbf{c},\mathbf{C}\right) = \sum_{pq}\Gamma_q^{p(I)}h_{pq} + \sum_{pqrs}\Gamma_{qs}^{pr(I)}\left(pq\,|rs\right)$$

Here, $\Gamma_q^{p(I)}$ and $\Gamma_{qs}^{pr(I)}$ are the one-and two-particle reduced electron density matrices for this state (labels p, q, r, s span the inactive and active subspaces):

$$\Gamma_q^{p(I)} = \left\langle \Psi_I^S \left| E_q^p \right| \Psi_I^S \right\rangle$$

$$\Gamma_{qs}^{pr(I)} = \frac{1}{2} \left\langle \Psi_I^S \left| E_q^p E_s^r - \delta_{qr} E_s^p \right| \Psi_I^S \right\rangle$$

The average energy is simply obtained from averaging the density matrices using arbitrary weights w_I that are user defined but are constrained to sum to unity.

$$\Gamma_q^{p(av)} = \sum_I w_I \Gamma_q^{p(I)}$$

$$\Gamma_{qs}^{pr(av)} = \sum_I w_I \Gamma_{qs}^{pr(I)}$$

$$\sum_I w_I = 1$$

3.13.3 Basic Usage

The most elementary input information which is always required for CASSCF calculations is the specification of the number of active electrons and orbitals.

```
%casscf
        nel  4  # number of active space electrons
        norb 6  # number of active orbitals
end
```

The CASSCF program in ORCA can average states of several multiplicities. The multiplicities are given as a list. For each multiplicity the number of roots should be specified:

```
%casscf mult   1,3 # here: multiplicities singlet and triplet

        nroots 4,2 # four singlets, two triplets
        end
```

If the symmetry handling in ORCA is enabled (`! UseSym`) each multiplicity block must have an irreducible representation assigned. Numbers corresponding to the "irrep" within a given symmetry are printed in the output of ORCA.

```
%casscf mult   1,3 # here: multiplicities singlet and triplet
        irrep  0,1 # here: irrep for each mult. block (mandatory!)
        nroots 4,2 # four singlets, two triplets
```

Several roots and multiplicities usually imply a state average CASSCF (SA-CASSCF) calculation. Note that the program by default chooses equal weights for the multiplicity blocks. Roots within a given block have equal weight. Users can define a custom weighting scheme for the multiplicity blocks and roots:

```
%casscf mult 1,3   # here: multiplicities singlet and triplet
        nroots 4,2  # four singlets, two triplets
        bweight 2,1 # singlets and triplets weighted 2:1
        weights[0] = 0.5,0.2,0.2,0.2 # singlet weights
        weights[1] = 0.7,0.3         # triplet weights
        end
```

The program automatically normalizes these weights such that the sum over all weights is unity. If convergence on an excited state is desired then the `weights[0]` array may look like `0.0,0.0,1.0` (this would optimize the orbitals for the third excited state. If several states cross during the orbital optimization this will ultimately cause convergence problems.

We note passing that the converged orbitals of the state averaged procedure are a compromise for the set of states. ORCA by default only prints the SA-CASSCF gradient norm. State-specific gradients are summarized at the end of the calculation with the keyword `PrintGState`.

```
%casscf
  ...
  printgstate true # optional printing of the state-specific orbital gradients
end
```

Orbital Optimization (2-step approach)

In the following we discuss the available options for the two-step ansatz, where orbital update and the CI problem are solved in sequence. A more powerful and slightly more expensive 1-step ansatz is described in section Section 3.13.3. We remark that a number of convergence problems can already be resolved changing the guess orbitals.

> **Warning**
>
> The following keywords are optional and should only be used facing severe convergence difficulties. Ideally, the default or `!TRAH` should be enough.

Aside from the perturbative SuperCI (`SuperCI_PT` - default),[430] several orbital optimization methods are implemented.

```
# Keywords to be used as Orbstep/Switchstep
SuperCI_PT  # perturbative SuperCI (first-order) - default
TRAH        # restricted-step augmented Hessian (second-order 1-step)
#
SuperCI     # SuperCI (first-order)
DIIS        # DIIS (first-order)
KDIIS       # KDIIS (first-order)
SOSCF       # approx. Newton-Raphson (first-order)
NR          # augmented Hessian Newton-Raphson
            # unfolded two-step procedure
            # - still not true second-order
```

The different convergers have different strengths. First-order method are cheap but typically require more iterations compared to second-order methods. When the gradient is far off from convergence the program uses the converger defined as `orbstep` while close to convergence the `switchstep` is used. The actual criteria for switchstep are defined with the keywords `SwitchConv` and `SwitchIter`.

```
%casscf
  OrbStep    SuperCI # or any other from the list above
  SwitchStep DIIS    # or any other from the list above

  SwitchConv 0.03 # gradient at which to switch
  SwitchIter 15   # iteration at which the switch takes place
                  # irrespective of the gradient

  MaxIter     75  # Maximum number of macro-iterations
end
```

Picking a convergence strategy, the program has to balance speed and robustness. The default strategy uses the `SuperCI_PT` as converger for `orbstep` and `switchstep`.[430] It is robust with respect to orbitals that are exactly doubly occupied or empty. Rotations with orbital close to this critical occupations can further be eliminated with the keyword `DThresh` (default=1e-6). However, the method is quite aggressive in the orbital optimization. In some cases, such as basis set projection or `PATOM` guess (intrinsic basis set projection), the program might pick a step-size that is too big. Then restricting the step-size via the keyword `MaxRot` (default=0.2) might be useful. The keywords `DThresh` and `MaxRot` described below are specific to `SuperCI_PT`. For many users, `MaxRot`

is less palpable than level shifting. Therefore, the present version allows level shifts as well. **In contrast to other convergers, level shifts are not needed and highly discouraged.** With the exception of `GradScaling` (*vide infra*), other damping techniques described further below do not apply to the `SuperCI_PT`.

```
# SuperCI_PT specific settings
MaxRot 0.05  # cap stepsize
DThresh 1e-6 # thresh for critical occupation
```

In case of convergence problems with the default settings, it is recommended to try the `!TRAH` option. Alternatively, the combination of `orbstep SuperCI` and `switchstep DIIS` in conjuction with a large level shift (2 Eh) often lead to immediate success. The proposed scheme typically requires more iterations. Moreoever, in contrast to the SuperCI(PT), the `SuperCI`, `DIIS` and `KDIIS` can struggle with active orbitals, that have an occupation of exactly 2.0 or 0.0! The `KDIIS` [6, 7] — based on perturbation theory — is an approximation to the regular `DIIS` procedure avoiding redundant rotations. Both DIIS schemes avoid linear dependencies in the expansion space.

```
MaxDIIS    15    # max. no of DIIS vectors to keep
DIISThresh 1e-7  # overlap criteria for linear dependency
```

The combination of `SuperCI` and `DIIS` (switchstep) is particularly suited to protect the active space composition. Adjusting the level shift will do the job. **Here, level shift is the single most important lever to control convergence**.

```
# default = dynamic level-shifting based on Ext-Act, Int-Act
ShiftUp      2.0   # static up-shift the virtual orbitals
ShiftDn      2.0   # static down-shift the internal orbitals
MinShift     0.6   # minimum separation subspaces
```

Level-shift is particularly important if the active, inactive and virtual orbitals overlap in their orbital energies. The energy separation of the subspaces is printed in the output. Ideally, the entries `Ext-Act` and `Act-Int` should be positive and larger than 0.2 Eh. This will help the program to preserve your active space composition throughout the iterations. If no shift is specified in the input, ORCA will choose a level-shift to guarantee an energy separation between the subspaces (`MinShift`).

```
E(CAS)=  -230.590325053 Eh DE=    -0.000798832
  --- Energy gap subspaces: Ext-Act = -0.244   Act-Int = -0.002
  --- current l-shift: Up(Ext-Act) = 0.54   Dn(Act-Int) = 0.30
```

In all of the implemented orbital optimization schemes the step-size correlates with the gradient-norm. A constant damping factor can be set with the keyword `GradScaling`. Note, damping and level shifting techniques are not recommended for the default converger (`SuperCI_PT`).

```
GradScaling  0.5   # constant damping in all steps
```

There are situations when the active space has been chosen carefully, but the initial gradient is still far off. To keep the "good" active space, we can suppress all rotation but the inactive-external ones until the gradient-norm is small enough to continue safely. The threshold can be set with `FreezeIE` keyword. Once the components of the gradient in the inactive-external direction have a weight of less than `FreezeIE`, all constraints are lifted. ORCA by default freezes active rotations if the total gradient norm is larger than 1.0 and the active rotations have a weight of less than 5%. The feature can be turned off setting the threshold to zero.

Similarly, rotations of the almost doubly occupied orbitals with the inactive orbitals can be damped using the threshold `FreezeActive`. Rotations of this type are damped as long as all their weight is smaller than `FreezeActive`. In contrast to the `ShiftDn`, it damps just the "troublesome" parts of internal-active rotations. This option applies to all of the orbital optimization schemes but the `SuperCI_PT`.

```
FreezeIE 0.4       # keep active space until int-ext rotation have
# a contribution of less than 40% to the ||g||
FreezeActive 0.03  # keep almost doubly occupied orbitals as long as
# their contribution is less than 3%  to the ||g||
```

If the calculation starts from a converged Hartree-Fock orbitals, the core orbitals should not change dramatically by the CASSCF optimization. Often trailing convergence is associated to rotations with low lying orbitals. Their

contribution to the total energy is fairly small. With the keyword `FreezeGrad` these rotations can be omitted from the orbital optimization procedure.

```
FreezeGrad   0.2  # omit  hitting a gradient norm ||g|| <0.2
```

The affected orbitals are printed at the startup of CASSCF.

```
FreezeGrad                          ... enabled if ||g|| is below 0.02
  Note Convergence can be signaled if the reduced gradient reaches GTol

  Last  frozen  orbital             ... 9
  First deleted orbital             ... 320
  Once rotations with core and deleted orbitals are stabilized they will be damped.
```

By default rotations with frozencore (or deleted virtuals) are not omitted. If the option FreezeGrad is active, the ratio with respect to the total gradient is printed.

```
||g|| =      0.001240414 Max(G)=    -0.000431747 Rot=319,1
  --- Option=FreezeGrad: ||g|| =      0.001081707
  = 87.21%
  Omitting frozencore elements
```

The DIIS based algorithms may sometimes converge slowly or "trail" towards the end. This is typically the region, where the augmented Hessian method (`NR`) and the cheaper quasi-Newton (`SOSCF`) perform better. Keep in mind that the Newton-Raphson is designed for optimization on a **convex surface** (Hessian is semidefinite). If the NR is switched on too early, there is a good chance that this condition is not fulfilled. In this case the program will complain about negative eigenvalues or diagonal elements of the Hessian as can be seen in the snippet below. The next optimization step is large and unpredictable. It is a wildcard that can get you closer to convergence or immediate divergence of the CASSCF procedure.

```
||g|| =      0.771376945 Max(G)=     0.216712933 Rot=140,53
  --- Orbital Update [        NR]
  Warning:  NEGATIVE diagonal element D(81,53)=    -4.733590
  Warning:  NEGATIVE diagonal element D(82,53)=    -4.737955
  ...
```

For larger system, the augmented Hessian equations are solved iteratively (NR iterations). The augmented Hessian is considered solved when the residual norm, $< r|r >$, is small enough. Aside from the overall CASSCF convergence, negative eigenvalues affect these NR iterations.

```
--- Orbital Update [        NR]
  AugHess Tolerance (auto): Tol= 1.00e-07
  AUGHESS-ITER   0: E=    -0.174480747 <r|r>=      0.558679452
  AUGHESS-ITER   1: E=    -0.308672359 <r|r>=      0.468254671
  AUGHESS-ITER   2: E=    -0.434272813 <r|r>=      0.286305469
  AUGHESS-ITER   3: E=    -0.439149451 <r|r>=      0.286514628
  AUGHESS-ITER   4: E=    -0.605787445 <r|r>=      0.191691955
  AUGHESS-ITER   5: E=    -0.607766529 <r|r>=      0.310450670
  AUGHESS-ITER   6: E=    -0.611674930 <r|r>=      0.141402593
  AUGHESS-ITER   7: E=    -0.623145299 <r|r>=      0.394505306
  AUGHESS-ITER   8: E=    -0.658410333 <r|r>=      0.166915094
  AUGHESS-ITER   9: E=    -0.790571374 <r|r>=      4.722929453
  AUGHESS-ITER  10: E=    -0.790590554 <r|r>=      4.716012014
  AugHess: No convergence in the Davidson procedure
  ...
```

There are a number of refined NR settings that influence the convergence behavior on a non-convex energy surface. We mention the keywords for completeness and disencourage from changing the default settings. If overall convergence cannot be changed due to negative eigenvalues, it is recommended to delay the NR switchstep (`switchconv`, `switchiter`). This will require some trial and error, since the curvature of the surface is *a priori* not know.

```
%casscf
  ...
  # NR specific settings:
  aughess
    Solver   0             # Davidson (default)
             1             # Pople (pure NR steps)
             2             # DIIS

    MaxIter 35             # max. no. of CI iters.
    MaxDim  35             # Davidson expansion space
    MaxDIIS  12            # max. number of DIIS vectors

    UseSubMatrixGuess true # diag a submatrix of the Hessian
                           # as an initial guess

    NGuessMat 512          # size of initial guess matrix (part of
                           # the Hessian exactly diagonalized)

    ExactDiagSwitch 512    # up to this dimension the Hessian
                           # is exactly diagonalized (small problems)

    PrintLevel 1           # amount of output during AH iterations
    Tol    1e-6            # convergence tolerance
    Compress true          # use compressed storage
    DiagShift 0.0          # shift of the diagonal elements of the
                           # Hessian

    UseDiagPrec true       # use the diagonal in updating
    SecShift 1e-4          # shift the higher roots in the Davidson
                           # secular equations

    UpdateShift 0.5        # shift of the denominator in the
                           # update of the AH coefficients
  end
end
```

Note

- Let us stress again: it is strongly recommended to first LOOK at your orbitals and make sure that the ones that will enter the active space are really the ones that you want to be in the active space! Many problems can be solved by thinking about the desired physical contents of the reference space before starting a CASSCF. A poor choice of orbitals results in poor convergence or poor accuracy of the results! Choosing the active orbitals always requires chemical and physical insight into the molecules that you are studying!
- Please try the program with default settings before playing with the more advanced options. If you encounter convergence problems, have a look into your output, read the warning and see how the gradient and energy evolves. Try `!TRAH`. Increasing `MaxIter` will not help in many cases.
- Be careful with keywords such as `!tightscf`, `!verytightscf` and so on. These keywords set higher integral thresholds, which is a good idea, but also tighten the CASSCF convergence thresholds. If you do not need a tighter energy convergence, reset the criteria in the casscf block using `ETol`. For many applications an energy convergence beyond 10^{-8} is unnecessary.

Incremental Fock

In general, convergence is strongly influenced by numerical noise, especially in the final iterations. One source of numerical noise is the incremental Fock build. Thus, it can help to enforce more frequent full Fock matrix formation.

```
ResetFreq 1 # reset frequency for direct SCF
```

If the orbital change in the active space is small, the active Fock matrix in ORCA is approximated using the density matrix from the previous cycle saving a second Fock matrix build. However, this approximation might also be a source of numerical instability. The threshold "SwitchDens" can be set to zero to enable the exact build. The program default starts with a rather large value (1e-2) and will reduce this parameter automatically when necessary.

```
switchdens  0.0001 # ~gtol * 0.1
```

Monitoring the active space

During the iterations, the **active orbitals are chosen to maximize the overlap** with active orbitals from the previous iterations. Maximizing the overlap does not make any restrictions on the nature of the orbitals. Thus initially localized orbitals will stay localized and ordered, which is sometimes a desired feature, e.g., in the density matrix renormalization group approach (DMRG). This feature is set with the keyword `ActConstraints` and is enabled by default (after the first 3 macro-iterations). For some orbital optimization procedures, such as the SuperCI, natural orbitals are more advantageous. Therefore, the `ActConstraints` can be turned off in favor of natural orbital construction (see below). If the keyword is not set by the user, ORCA picks the best choice for the given orbital optimization step.

```
ActConstraints  0 # no checks and no changes
                1 # maximize overlap of active orbitals and check sanity (default␣
↪for DIIS)
                2 # make natural orbitals in every iteration (default SuperCI)
                3 # make canonical orbitals in every iteration
                4 # localize orbitals
```

In addition to maximizing the overlap, **`"ActConstraints 1"` checks if the composition of the active space has changed**, i.e., an orbital has been rotated out of the active space. In this case, ORCA aborts and stores the last valid set of orbitals. Below is an example error message.

```
--- Orbital Update [      DIIS]
  --- Failed to constrain active orbitals due to rotations:
  Rot( 37, 35) with OVL=0.960986
  Rot( 38, 34) with OVL=0.842114
  Rot( 43,104) with OVL=0.031938
```

In the snippet above, the active space ranges from 37-43. The program reports that orbitals 37,38 and 43 have changed their character. The overlap of orbital 43 (active) with the previous set of active orbitals is just 3% and the program aborts. There are a number of reasons, why this happens in the calculation. If this error occurs constantly with the same orbitals, it is worthwhile to inspect the rotating partner orbitals (visualize). It might be sign that the active space is not balanced and should be extended. In many instances changing the level-shift or lowering switchconv is sufficient to protect the active space. In some cases, turning off the sanity check (`"ActConstraints 0"`) and re-rotating orbitals will bring CASSCF closer to convergence. Some problems can be avoided by a better design of the calculation. The CASSCF tutorial elaborates on the subject in more detail.

There are situations such as parameter scans, where "actconstraints" is counter-productive and should be disabled. In other words, we want to allow changes in the active space composition. As an example, consider the rotation of ethylene around its double-bond represented by a CAS(2,2). Although the active space consists of the bonding and anti-bonding orbitals π-orbitals, their composition in terms of atomic orbitals changes from the eclipsed to the staggered conformation. Depending on the actual input settings (orbstep and number of scan points), this might trigger an abort.

Orbital Optimization (1-step approach): Robust Convergence with TRAH-CASSCF

The restricted-step second-order converger TRAH is now also available for both state-specific and state-averaged CASSCF calculations.[121] To activate TRAH for your CASSCF calculation, you just need to add `!TRAH` in one of the simple input lines and add an auxiliary basis.

```
# Auxiliary basis of type /C needed for !TRAH
! TRAH Def2-SVP Def2-SVP/C TightSCF

%casscf
  # define CAS(6,6)
  nel 6
  norb 6

  # two lowest singlets states
  mult 1
  nroots 2
end

*xyz 0 1
  N   0.0   0.0   0.0
  N   0.0   0.0   1.1
end
```

In most cases, there is no need to play with any input parameters. The only exception is the choice of active molecular orbital representations that can have a significant impact on the convergence rate for spin-coupled systems. As can be seen from Fig. Fig. 3.13, for such calculations localized active orbitals perform best. In any other case, the natural orbitals (default) should be employed.

Fig. 3.13: SXPT and TRAH error convergence using different choices for the active-orbital basis.

Possible input options for the *active-orbital basis* are

```
%casscf
 ActConstraints Natural   # default
end
```

Active Orbitals	Meaning
NatOrbs	Keeps the one-electron density matrix (1-RDM) diagonal. **Default**
LocOrbs	A Foster-Boys localization of active MOs is performed in every macro iteration. This is recommended for **spin-coupled systems**.
Unchanged	The active MO basis is not changed. Primarily debug option.
CanonOrbs	Keeps the total active-MO Fock matrix diagonal. Experimental option.

Note that, in contrast to the SCF program, there is **no** `AutoTRAH` feature for CASSCF yet. The TRAH feature has to be requested explicitly in the input.

Using the RI Approximation

A significant speedup of CASSCF calculations on larger molecules can be achieved with the RI, RI-JK and RIJ-COSX approximations. [430] There are two independent integral generation and transformation steps in a CASSCF procedure. In addition to the usual Fock matrix construction, that is central to HF and DFT approaches, more integrals appear in the construction of the orbital gradient and Hessian. The latter are approximated using the keyword `trafostep RI`, where an auxiliary basis (/C or the more accurate /JK auxiliary basis) is required. Note that auxiliary basis sets of the type /J are not sufficient to fit these integrals. If no suitable auxiliary basis set is available, the `AutoAux` feature might be useful.[59] We note passing, that there are in principle three distinguished auxiliary basis slots, that can be individually assigned in the `%basis` block (section Section 2.7). As an example, we recompute the benzene ground state example from section Section 3.13.6 with a CAS(6,6).

```
! SV(P) def2-svp/C
! moread
%moinp "Test-CASSCF-Benzene-2.mrci.nat"

# Commented out: Detailed settings of the auxiliary basis in the %basis block,
#                where the AuxC slot is relevant for the option TrafoStep RI.
# %basis
# auxC "def2-svp/C"
# end

%casscf
  # define CAS(6,6)
  nel    6
  norb   6

  # singlet ground state
  mult   1
  nroots 1

  trafostep ri # enables the RI approximation
end
```

The energy of this calculation is `-230.590328 Eh` compared to the previous result `-230.590271 Eh`. Thus, the RI error is only 0.06 mEh which is certainly negligible for all intents and purposes. With the larger `/JK` auxiliary basis the error is typically much smaller (0.02 mEh in this example). Even if more accurate results are necessary, it is a good idea to pre-converge the CASSCF with RI. The resulting orbitals should be a much better guess for the subsequent calculation without RI, and thus save computation time.

The `TrafoStep RI` only affects the integral transformation in CASSCF calculations while the Fock operators are still calculated in the standard way using four index integrals. In order to fully avoid any four-index integral evaluation, you can significantly speed up the time needed in each iteration by specifying `!RIJCOSX`. The keyword implies `TrafoStep RI`. The COSX approximation is used for the construction of the Fock matrices. In this case, an additional auxiliary basis (**/J** auxiliary basis) is mandatory.

```
! SV(P) def2-svp/C RIJCOSX def2/J
! moread
%moinp "Test-CASSCF-Benzene-2.mrci.nat"

# Commented out: Detailed settings of the auxiliary basis in the %basis block,
#                where the AuxJ and AuxC slot are mandatory.
# %basis
# auxJ "def2/J"
# auxC "def2-svp/C"
# end

%casscf
  # define CAS(6,6)
  nel    6
  norb   6

  # singlet ground state
  mult   1
  nroots 1
end
```

The speedup and accuracy is similar to what is observed in RHF and UHF calculations. In this example the `RIJCOSX` leads to an error of 1 mEh. The methodology performs better for the computation of energy differences, where it profits from error cancellation. The RIJCOSX is ideally suited to converge large-scale systems. Note that for large calculations the integral cut-offs and numerical grids should be tightened. See section Section 2.8.4 for details. With a floppy numerical grid setting the accuracy as well as the convergence behavior of CASSCF deteriorate. The RIJK approximation offers an alternative ansatz. The latter is set with `!RIJK` and can also be run in conventional mode (`conv`) for additional speed-up. With `conv`, a **single auxiliary basis must be provided** that is sufficiently larger to approximate the Fock matrices as well the gradient/Hessian integrals. In direct mode an additional auxiliary basis set can be set for the `AuxC` slot.

```
! SV(P) RIJK  def2/JK

# Commented out: Detailed settings of the auxiliary basis in the %basis block,
#                where only the auxJK slot must be set.
# %basis
# auxJK "def2/JK" # or "AutoAux"
# end
```

The RIJK methodology is more accurate and robust for CASSCF, e.g., here the error is just 0.5 mEH.

Final Orbitals: Canonicalization Choices

Once the calculation has converged, ORCA will do a final macro-iteration, where the orbital are "finalized". For complete active spaces (CAS), these transformations do not alter the final energy and wave function. Note, that solutions from approximate CAS-CI solvers such as the ICE approach or the DMRG ansatz are affected by the final orbital transformation. The magnitude depends on the truncation level (e.g. `TGen`, `TVar` and `MaxM`) of the approximated wave function. The default final orbitals are canonical in the internal and external space with respect to state-averaged Fock operator. The active orbitals are chosen as natural orbitals. Other orbital choices are equally valid and can be selected for the individual subspaces.

```
#internal space
IntOrbs CanonOrbs # canonical
        LocOrbs   # localized
        unchanged # no changes

        # partner orbitals for the active space based
        # on various concepts
        PMOS      # based on orthogonalization tails.
```

(continues on next page)

(continued from previous page)

```
        OSZ       # based on oszillator orbital
        DOI       # based on differential overlap

#external space
ExtOrbs CanonOrbs # canonical
        LocOrbs   # localized
        unchanged # no changes

        # partner orbitals for the active space based
        # on various concepts
        PMOS      # based on orthogonalization tails.
        OSZ       # based on oszillator orbital
        DOI       # based on differential overlap
        DoubleShell # based on the shell and angular momentum
                    # of the highest active orbitals, the first few
                    # virtual orbitals correspond to the doubled-shell.
                    # All other virt. orbitals are canonicalized.
                    # For 3d-metal complexes, these are the 4d orbitals!
                    # For 4d-metal complexes, these are the 5d orbitals!
                    # And so on...
#active space
ActOrbs NatOrbs   # natural
        CanonOrbs # canonical
        LocOrbs   # localized
        unchanged # no changes
        dOrbs     # purify metal d-orbital and call the AILFT
        fOrbs     # purify metal f-orbital and call the AILFT
        SDO       # Single Determinant Orbitals: this is only possible if the
                  # active space has a single hole or a single electron.
                  # SDOs are then the unique choice of orbitals that simultaneously
                  # turns each CASSCF root into a single determinant.
```

SDOs are specific for the active orbital space.[432] The set of options (`PMOS, OSZ, DOI, DoubelShell`) are specific for the inactive and external space. They aim to assist the extension of the current active space. All four options, re-design the first `NOrb` (number of active orbitals) next to the active space, while the remaining orbitals of the same subspace are canonical. The re-designed orbital are based on different concepts.

- `PMOS` generates the bonding / anti-bonding partner orbitals for the chosen active space. It is based on the orthogonalization tail of the active orbitals.
- `OSZ` generates a single orbital for each active orbital, that maximizes the dipole-dipole interaction.
- `DOI` follows the same principle as OSZ, but the differential overlap is maximized instead.
- `DoubleShell` is specific to the external space. The highest active MO or `DoubleShellMO` is analyzed. A set of orbitals with the same angular momentum but larger radial distribution is generated.

Optionally, the four options above can be supplemented with a reference MO using the keyword `RefMO/DoubleShellMO`. The presence of `RefMO/DoubleShellMO` changes the default behavior. In case of `PMOS`, `OSZ` and `DOI`, all orbitals of the given subspace are chosen to maximize a single objective function with respect to the reference MO (must be active). This contrasts the default settings, where for each active MO an objective function is maximized and a single "best" orbital is picked. In other words, in the default setting, each active orbital has a corresponding "best" orbital in the selected subspace neighboring the active space.

```
RefMO 17         # MO with number 17 (default =-1)
DoubleShellMO 17 # MO with number 17 (default =-1)
```

The aforementioned options are aids and the resulting orbitals should be inspected prior extension of the active space. In particular the `PMOS` option is useful in the context of transition metal complexes to find suitable Ligand based orbitals. There are more options (`dorbs, forbs, DoubleShell`), that are specifically designed for coordination chemistry. A more detailed description is found in the CASSCF tutorial that supplements the manual.

If the active space consists of a single set of metal d-orbitals, natural orbitals may be a mixture of the d-orbitals.

The active orbitals are remixed to obtain "pure" d-orbitals (ligand field orbitals) if the `actorbs` is set to `dorbs`. The same holds for f-orbitals and the option `forbs`. Furthermore, the keyword `dorbs` automatically triggers the *ab initio* ligand field analysis (AILFT).[433, 434]The approach has been reported in a number of applications.[435, 436, 437, 438, 439, 440, 441, 442] Note that the actual representation depends on the chosen axis frame. It is thus recommended to align the molecule properly. For more details on the AILFT approach, we refer to the AILFT section Section 3.13.16 , the original paper and the CASSCF tutorial, where examples are shown.

CI Solvers (CSFCI, ACCCI, ..., ICE)

The CI-step default setting is CSF based and is done in the present program by generating a partial "formula tape" which is read in each CI iteration. The tape may become quite large beyond several hundred thousand CSFs which limits the applicability of the CASSCF module. The accelerated CI (ACCCI) has the same limitations, but uses a slightly different algorithm that handles multi-root calculations much more efficiently.

Larger active spaces are tractable with the *DMRG* approach or *the iterative configuration expansion (ICE)* developed in our own group.[443, 444] DMRG and ICE return approximate full CI results. The maximum size of the active space depends on the system and the required accuracy. Active spaces of 10–40 orbitals should be feasible with both approaches. The CASSCF tutorial covers examples with ACCCI and ICE as CI solvers.

```
%casscf
  CIStep CSFCI # CSF based CI (default)
         ACCCI  # CSF based CI solver with faster algorithm for multi-root␣
↪calculations
         ICE    # CSF based approximate CI -> ICE/CIPSI algorithm
         DMRGCI # density matrix renormalization group approach instead of the CI
end
```

In the ICE approach, the computation of the coupling coefficients is time-consuming and by default repeated in every macro-iteration. To avoid the reconstruction, it is recommended to once generate a coupling coefficient library (`cclib`) and to use it for all of your ICE calculations. The details of the methodology and the `cclib` are described in the ICE section Section 3.14.

Detailed settings for the conventional CI solvers (CSFCI, ACCCI, ICE) can be controlled in a sub-block.

```
%casscf
...
  # CI specific settings
  ci
    MaxIter 64     # max. no. of CI iters.
    MaxDim  10     # Davidson expansion space = MaxDim * NRoots
    NGuessMat 512  # Initial guess matrix: 512x512
    PrintLevel 3   # amount of output during CI iterations
    ETol   1e-10   # default 0.1*ETol in CASSCF
    RTol   1e-10   # default 0.1*ETol in CASSCF
    TGen   1e-4    # ICE generator thresh
    TVar   1e-11   # ICE selection thresh, default = TGen*1e-7

    TPrint  1e-3  # Thresh to print the CI Coefficients / CI weights
    PrintWF 0     # (default) prints only the CFGs
            csf   # Printing of the wave function in the basis of CSFs
            det   # Printing of the wave function in the basis of Determinants

    # ICE/CIPSI specific settings
    # See ICE section of the manual:
    CIPSIType  0   # 0: CFG based (default)
                   # 1: CSF based
    TVar     -1e.7 # variational threshold. negative means 1e-7*TGen
    TGen      1e-4 #  generator threshold
```

(continues on next page)

(continued from previous page)

```
  end
end
```

The `CI-step` DMRGCI interfaces to the **BLOCK** program developed in the group of G. K.-L. Chan [445, 446, 447, 448]. A detailed description of the **BLOCK** program, its input parameters, general information and examples on the density matrix renormalization group (DMRG) approach, are available in the section *Density Matrix Renormalization Group (DMRG)* of the manual.

The implementation of DMRG in **BLOCK** is fully spin-adapted. However, spin-densities and related properties are not available in the current version of the **BLOCK** code. To start a DMRG calculation add the keyword "`CIStep DMRGCI`" into a regular CASSCF input. ORCA will set default parameters and generate and input for the **BLOCK** program. In general, DMRG is not invariant to rotation in the active space. The program by **default** will run an automatic ordering procedure (`Fiedler`). More and refined options can be set in the `dmrg` sub-block of CASSCF — see section Section 3.15 for a complete list of keywords.

```
%casscf
  nel  8
  norb 6
  mult 3
  CIStep DMRGCI

  # Detailed settings
  dmrg
    # more/refined options
    ...
  end
end
```

It is highly recommended to start the calculation with split-localized orbitals. Any set of starting orbitals (gbw file) can be localized using the `orca_loc` program. Typing `orca_loc` in the shell will return a small help-file with details on how to setup an input for the localization. Examples for DMRG including the localization are in the corresponding section of the manual *Density Matrix Renormalization Group (DMRG)*. The utility program `orca_loc` is documented in section Section 9.2.5. Split-localization refers to an independent localization of the internal and virtual part of the desired active orbitals.

Model Spaces: CAS, RAS and ORMAS

In the MCSCF approach, the CI- and orbital coefficients are optimized. The complete active space (CAS) is a specific models space, where the CI consists of all possible CSFs, that have N active electron in M active orbitals (i.e. CAS(N,M)). This is naturally the default for the `%casscf` module. However, the module is general enough to define arbitrary CSFs using their configurations (CFGs). The latter refer to the orbital occupation of the CSFs.

> **Warning**
>
> The orbital optimization is carried out omitting the active-active rotation, which might be non-negligible. Hence, the energy is sensitive to canonicalization choices (see `actorbs`,`actconstraints`).

Arbitrary References

With the `refs` sub-block, arbitrary CFGs can be selected to construct the CI space. All CSFs within a given CFG are accounted for. This is for example useful in the *context of X-ray spectroscopy*.

As simple example, lets take the benzene molecule from section Section 3.13.6 and restrict the CI space to the RHF determinant

```
! SV(P)
! moread
%moinp "Test-CASSCF-Benzene-2.mrci.nat"

%casscf
  # define arbitrary CI space using the CFG input mask
  refs
    {2 2 2 0 0 0} # CFG with the lowest orbitals being doubly occupied.
                  # Arbitrary CFGs can be added in this block.
                  # The number of electrons and orbitals
                  # must match the active space defined below.

  end

  # define active space
  nel    6
  norb   6

  # define singlet ground state
  mult 1
  nroots 1
end
```

Restricted Active Space (RAS)

In the RAS model space, the active space is further partitioned into RAS1, RAS2 and RAS3. The input `RAS(Nel: R1 H/R2/ R3 P)` requires 6 parameters: 'Nel' refers to the total number of electrons in the system. 'R1/R2/R3' define the number orbitals in RAS1/RAS2/RAS3. 'H' defines the number of holes in the RAS1 CFGs, whereas 'P' the number of particles in the RAS3. With the restrictions above, all possible CFGs are constructed. RAS is particularly useful to define *core-excited states*.

```
%casscf
 # define the active space (mandatory)
 nel 11
 norb 8
 # define the actual RAS partitioning
 refs
   RAS(11:3 1/5/0 0) # (Nel: NRAS1 MaxHoles / NRAS2 / NRAS3 MaxParticles)
 end
 ...
end
```

Occupation Restricted Multiple Active Spaces (ORMAS)

A more general partitioning is possible with the ORMAS model space, which consists of multiple active spaces (up to 25). Each active space is defined with the number of orbitals('m') and the number of electrons ranging from 'min' to 'max'. With the input mask `ORMAS(nel: m1 min1 max1, m2 min2 max2,...)`, the active spaces are defined such that a total of 'nel' electrons are distributed among all the sub-space. The first sub-space (m1, min1, max1) consists of 'm1' orbitals with electrons from 'min1' to 'max1'. The subsequent parameters define the next sub-space (m2,min2,max2) and so on. Note that the standard CASSCF parameters `nel` and `norb` are overwritten with the parameters from the ORMAS input mask.

```
%casscf
 # define the active space (redundant)
 nel 10
 norb 15
 # define the ORMAS partitioning (overwritting nel/norb)
 refs
  ORMAS(10: 3 4 6 / 2 0 4/ 10 0 2) # 10 electron distributed to:
                                   # first space  = 3 orbitals with 4-6 electrons
                                   # second space = 2 orbitals with 0-4 electrons
                                   # third space  = 10 orbitals with 0-2 electrons
 end
 ...
end
```

The RASCI and ORMAS Module

ORCA features a rather flexible module that is designed to handle rather flexible general CI expansions. This is not meant to be in competition with the individually selecting and uncontracted MRCI or the internally contracted AUTOCI program or the iteratively refined ICE program. Rather, this is an attempt to create a simple and flexible module that allows expert users to play with various general CI expansions using moderate orbital spaces. This is not a module that can be used on large molecules or systems with hundreds let alone thousands of orbitals. One can use it for larger molecules only in a setting where there is a limited orbital window.

Importantly, the module can accept individual configurations or CSFs as references, RAS reference and also ORMAS references as detailed below. In addition to these flexible reference space, the program can also perform excitations of various degrees on these references. The module is connect with the QDPT driver for the calculation of magnetic properties and the OPA driver for the calculation of optical spectra.

```
! RHF cc-PVDZ VeryTightSCF PModel

# AVAS guess for sigma/sigma* active space
%scf
 avas
  system
   shell  2, 2
   l      1, 1
   m_l    0, 0
   center 0, 1
  end
 end
end

# casscf single bond correlation
%casscf nel    2
        norb   2
        irrep  0
        mult   1
        nroots 1
        cistep 0
        trafostep exact
```

(continues on next page)

(continued from previous page)

```
        end

%rasci refs RAS (14: 6 2 / 2 / 18 2)
    # this is MRCISD
    # MRCISDT would be RAS(14: 6 3 / 2 / 18 3)
    # no of corr. el.: RAS1 norb nholes/ RAS2 nel/RAS3 norb nel)
    #ORMAS(14: 6 10 12/ 2 0 2/ 18 0 2)
    #would be identical. ORMAS can be used with up to 25
    # subspaces all given as norb nelmin nelmax
    # Indiviual CFGS { 2 2 2 2 2 2 1 1 0 } would be possible
      end

  # irrep lists can be used as in CASSCF
  # irrep   0
   mult   1,3    # multiplicity lists as in CASSCF
   nroots 5,5    # number of root lists as in CASSCF
   cistep accci  # accelerated CI as in CASSCF
          #csfci : CI in CSFs as in CASSCF
          #detci : CI in determinants as in ICE
          #treecsf: CI in CSFs as in ICE

   ExcLevel 0              # excitation level on top of refs
   RejectInvalidRefs false # reject references of invalid spin and
                           # space symmetry before excitations
                           # as in MRCI
   douv true               # optical spectra (UV,CD,MCD)
   rel dosoc true end      # Magnetic properties
  end

* xyz 0 1
F 0 0 0
F 0 0 1.457513043738
*
```

Relevant Papers:

1. Ivanic, Joseph. Direct configuration interaction and multiconfigurational self-consistent-field method for multiple active spaces with variable occupations. I. Method. *The Journal of Chemical Physics*, **2003**, 119 (18), 9364–9376. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/119/18/9364/19131699/9364_1_online.pdf, DOI: 10.1063/1.1615954.

3.13.4 Keywords

Simple input keywords for CASSCF are given in Table 3.47. All of them require the CASSCF block as minimal input, the options for which are given below.

Table 3.47: Simple input keywords for CASSCF

Keyword	Description
DMRG	Sets DMRG as "CIStep" in CASSCF
NEVPT2	SC NEVPT2
SC-NEVPT2	SC-NEVPT2 same as NEVPT2
RI-NEVPT2	SC-NEVPT2 with the RI approximation
FIC-NEVPT2	FIC-NEVPT2 aka PC-NEVPT2
DLPNO-NEVPT2	FIC-NEVPT2 in the framework of DLPNO
CASPT2	FIC-CASPT2
RI-CASPT2	FIC-CASPT2 with the RI approximation
DCD-CAS(2)	2nd order Dynamic Correlation Dressed CAS
RI-DCD-CAS(2)	2nd order Dynamic Correlation Dressed CAS with RI approximation

Warning

Please always try the default convergence settings before tweaking the convergence manually!

```
%casscf

  PrintLevel 3 # amount of printing

  # ----------------------------------------------------
  # definition of active space
  # (mandatory)
  # ----------------------------------------------------
  nel  6  # number of active space electrons
  norb 6  # number of active orbitals

  # ----------------------------------------------------
  # incomplete model spaces
  # (optional)
  # ----------------------------------------------------
  # define arbitrary CI space using the CFG input mask
  refs
    # arbitrary CFGs.
    {2 2 2 0 0 0}
    # RAS(Nel: NRAS1 MaxHoles / NRAS2 / NRAS3 MaxParticle)
    RAS(6:2 1/2/2 1)
    #ORMAS partitioning (overwritting nel/norb)
    ORMAS(10: 3 4 6 / 2 0 4/ 10 0 2) # 10 electron distributed to:
                                     # first space  = 3 orbitals with 4-6 electrons
                                     # second space = 2 orbitals with 0-4 electrons
                                     # third space  = 10 orbitals with 0-2␣
↪electrons
  # ----------------------------------------------------
  # definition of states
  # (mandatory)
  # ----------------------------------------------------
  mult   1,3 # here: multiplicities singlet and triplet
  irrep  0,1 # here: irrep for each mult. block (mandatory!)
  nroots 4,2 # four singlets, two triplets

  # ----------------------------------------------------
  # State-Averaging
  # (optional)
  # ----------------------------------------------------
  bweight 2,1 # singlets and triplets weighted 2:1
  weights[0] = 0.5,0.2,0.2,0.2 # singlet weights
  weights[1] = 0.7,0.3         # triplet weights
  printgstate true # optional printing of the state-specific orbital gradients


  # ----------------------------------------------------
  # final orbital option (canonicalization)
  # (optional)
  # ----------------------------------------------------
  #internal space
  IntOrbs CanonOrbs # canonical - default
          LocOrbs   # localized
          unchanged # no changes

          # partner orbitals for the active space based
          # on various concepts
          PMOS      # based on orthogonalization tails.
```

(continues on next page)

(continued from previous page)

```
        OSZ       # based on oszillator orbital
        DOI       # based on differential overlap
#external space
ExtOrbs CanonOrbs # canonical - default
        LocOrbs   # localized
        unchanged # no changes

        # partner orbitals for the active space based
        # on various concepts
        PMOS      # based on orthogonalization tails.
        OSZ       # based on oszillator orbital
        DOI       # based on differential overlap
        DoubleShell # based on the shell and angular momentum
                    # of the highest active orbitals, the first few
                    # virtual orbitals correspond to the doubled-shell.
                    # All other virt. orbitals are canonicalized.
                    # For 3d-metal complexes, these are the 4d orbitals!
                    # For 4d-metal complexes, these are the 5d orbitals!
                    # And so on...
#active space
ActOrbs NatOrbs   # natural - default
        CanonOrbs # canonical
        LocOrbs   # localized
        unchanged # no changes
        dOrbs     # purify metal d-orbital and call the AILFT
        fOrbs     # purify metal f-orbital and call the AILFT
        LMORBS    # calls extended AI-LFT
        SDO       # Single Determinant Orbitals: this is only possible if the
                  # active space has a single hole or a single electron.
                  # SDOs are then the unique choice of orbitals that
                  # simultaneously turns each CASSCF root into a
                  # single determinant.

# PMO         reference orbitals
RefMO 17         # MO with number 17 (default =-1)
# DoubleShell reference orbital
DoubleShellMO 17 # MO with number 17 (default =-1)

# ----------------------------------------------------
# NTO / NDO printing
# ----------------------------------------------------
DoNTO           #Generate Natural Transition Orbitals
NTOStates 1,2,3 #States to consider for NTO analysis. If empty, all are done
NTOThresh 1e-4  #Threshold for printing occupation numbers
DoNDO           #Generate Natural Difference Density Orbitals
NDOStates 1,2,3 #States to consider for NDO analysis. If empty, all are done

# ----------------------------------------------------
# RI approximation of the gradient/ Hessian integrals
# (optional)
# ----------------------------------------------------
TrafoStep RI # enabled RI approximation with /C aux basis

# ----------------------------------------------------
# FOCK options
# ----------------------------------------------------
ResetFreq 15       # reset frequency for direct SCF
switchdens  0.0001 # approximate active Fock when density is considered unchanged

# ----------------------------------------------------
# CI settings
```

(continues on next page)

(continued from previous page)

```
# ----------------------------------------------------
CIStep CSFCI  # CSF based CI (default)
       ACCCI  # CSF based CI solver, faster algorithm for multi-root calculations
       ICE    # CSF based approximate CI -> ICE/CIPSI algorithm
       DMRGCI # density matrix renormalization group approach instead of the CI
# CI specific settings
CI
  MaxIter 64     # max. no. of CI iters.
  MaxDim  10     # Davidson expansion space = MaxDim * NRoots
  NGuessMat 512  # Initial guess matrix: 512x512
  PrintLevel 3   # amount of output during CI iterations
  ETol   1e-10   # default 0.1*ETol in CASSCF
  RTol   1e-10   # default 0.1*ETol in CASSCF
  TGen   1e-4    # ICE generator thresh
  TVar   1e-11   # ICE selection thresh, default = TGen*1e-7

  TPrint  1e-3  # Thresh to print the CI Coefficients / CI weights
  PrintWF 0     # (default) prints only the CFGs
          csf   # Printing of the wave function in the basis of CSFs
          det   # Printing of the wave function in the basis of Determinants

  # ICE/CIPSI specific settings
  # See ICE section of the manual:
  ICEType  CFGs #  CFG based (default)
           CSFs #  CSF based
           DETs #  determinant based
  TVar     -1e.7 # variational threshold. negative means 1e-7*TGen
  TGen      1e-4 #  generator threshold
end

# ----------------------------------------------------
# Numerical gradient state-selection
# ----------------------------------------------------
imult 0 # multiplicity block counting from zero
iroot 0 # root for given imult counting from zero

# ----------------------------------------------------
# orbital optimization
# (optional)
# ----------------------------------------------------

GTol  1e-3  # convergence criteria for ||g||
ETol  1e-8  # convergence criteria for energy

# The following options allowed for 'orbstep'/'switchstep'
# SuperCI_PT - perturbative SuperCI (first-order) - default 2-step
# TRAH       - restricted-step augmented Hessian (second-order 1-step)
# SuperCI    - SuperCI (first-order)
# DIIS       - DIIS (first-order)
# KDIIS      - KDIIS (first-order)
# SOSCF      - approx. Newton-Raphson (first-order)
# NR         - augmented Hessian Newton-Raphson
#            - unfolded two-step procedure (not true second order)

orbstep SuperCI_PT    # default
switchstep SuperCI_PT # default after ||g|| < 'SwitchConv'
SwitchConv 0.03 # gradient norm ||g|| at which to switch
SwitchIter 15   # iteration at which the switch takes place
                # irrespective of the gradient

MaxIter     75  # Maximum number of macro-iterations
```

(continues on next page)

(continued from previous page)

```
  # level shifts / damping etc
  ShiftUp      2.0   # static up-shift the virtual orbitals
  ShiftDn      2.0   # static down-shift the internal orbitals
  MinShift     0.6   # minimum separation subspaces

  # monitoring the active space
  ActConstraints  0 # no checks and no changes
          1 # maximize overlap of active orbitals and check sanity (default for␣
↪DIIS)
          2 # make natural orbitals in every iteration (default SuperCI)
          3 # make canonical orbitals in every iteration
          4 # localize orbitals

  # 'SuperCI_PT' specific settings
  MaxRot 0.05  # cap stepsize
  DThresh 1e-6 # thresh for critical occupation

  # 'TRAH' specific settings
  # see 'TRAH' section of manual.

  # DIIS/KDIIS/SuperCI_PT specific
  MaxDIIS    15    # max. no of DIIS vectors to keep
  DIISThresh 1e-7  # overlap criteria for linear dependency

  # 'NR' specific settings:
  aughess
    Solver   0             # Davidson (default)
             1             # Pople (pure NR steps)
             2             # DIIS

    MaxIter 35             # max. no. of CI iters.
    MaxDim  35             # Davidson expansion space
    MaxDIIS  12            # max. number of DIIS vectors

    UseSubMatrixGuess true # diag a submatrix of the Hessian
                           # as an initial guess

    NGuessMat 512          # size of initial guess matrix (part of
                           # the Hessian exactly diagonalized)

    ExactDiagSwitch 512    # up to this dimension the Hessian
                           # is exactly diagonalized (small problems)

    PrintLevel 1           # amount of output during AH iterations
    Tol    1e-6            # convergence tolerance
    Compress true          # use compressed storage
    DiagShift 0.0          # shift of the diagonal elements of the
                           # Hessian

    UseDiagPrec true       # use the diagonal in updating
    SecShift 1e-4          # shift the higher roots in the Davidson
                           # secular equations

    UpdateShift 0.5        # shift of the denominator in the
                           # update of the AH coefficients
  end

  # more dampng options restricted to a few selected 'orbstep'/'switchstep'
  FreezeIE 0.4       # keep active space until int-ext rotation have
                     # a contribution of less than 40% to the ||g||
```

(continues on next page)

(continued from previous page)

```
  FreezeActive 0.03  # keep almost doubly occupied orbitals as long as
                     # their contribution is less than 3%  to the ||g||
  FreezeGrad   0.2   # omit  hitting a gradient norm ||g|| <0.2
  GradScaling  0.5   # constant damping in all steps

  # -------------------------------------------------------
  # one photon spectroscopy
  # QDPT properties  (rel subblock)
  # DMRG
  # NEVPT2
  # DCD-CAS
  # MR-ADC0
  # AI-LFT
  # TRAH 1-step orbital optimization
  # -------------------------------------------------------
  # See respective manual section for details
```

3.13.5 A simple Example: Be Ground State

One standard example of a multireference system is the Be atom. Let us run two calculations, a standard closed-shell calculation ($1s^2 2s^2$) and a CASSCF(2,4) calculation which also includes the ($1s^2 2s^1 2p^1$) and ($1s^2 2s^0 2p^2$) configurations.

```
! TZVPP TightSCF
* xyz 0 1
Be 0 0 0
*
```

This standard closed-shell calculation yields the energy `-14.56213241 Eh`. The CASSCF calculation

```
! TZVPP TightSCF

%casscf
        # defining CAS(2,4)
        nel  2
        norb 4

        # ground state singlet:
        mult 1   # define the multiplicity.
        nroots 1 # define the number of roots for the mult=1
end

* xyz 0 1
Be 0 0 0
*
```

yields the energy `-14.605381525 Eh`. Thus, the inclusion of the 2p shell results in an energy lowering of 43 mEh which is considerable. The CASSCF program also prints the composition of the wave function:

```
---------------------------------------------
CAS-SCF STATES FOR BLOCK  1 MULT= 1 NROOTS= 1
---------------------------------------------

ROOT   0:  E=     -14.6053815294 Eh
      0.90060 [     0]: 2000
      0.03313 [     4]: 0200
      0.03313 [     9]: 0002
      0.03313 [     7]: 0020
```

This information is to be read as follows: The lowest state is composed of 90% of the configuration which has the active space occupation pattern 2000 which means that the first active orbital is doubly occupied in this configuration while the other three are empty. The MO vector composition tells us what these orbitals are (ORCA uses natural orbitals to canonicalize the active space).

```
                      0         1         2         3         4         5
                 -4.70502  -0.27270   0.11579   0.11579   0.11579   0.16796
                  2.00000   1.80121   0.06626   0.06626   0.06626   0.00000
                 --------  --------  --------  --------  --------  --------
 0 Be s            100.0     100.0       0.0       0.0       0.0     100.0
 0 Be pz             0.0       0.0      13.6       6.1      80.4       0.0
 0 Be px             0.0       0.0       1.5      93.8       4.6       0.0
 0 Be py             0.0       0.0      84.9       0.1      15.0       0.0
```

Thus, the first active space orbital has occupation number 1.80121 and is the Be-2s orbital. The other three orbitals are 2p in character and all have the same occupation number 0.06626. Since they are degenerate in occupation number space, they are arbitrary mixtures of the three 2p orbitals. It is then clear that the other components of the wave function (each with 3.31%) are those in which one of the 2p orbitals is doubly occupied.

How did we know how to put the 2s and 2p orbitals in the active space? The answer is – WE DID NOT KNOW! In this case it was "good luck" that the initial guess produced the orbitals in such an order that we had the 2s and 2p orbitals active. **IN GENERAL IT IS YOUR RESPONSIBILITY THAT THE ORBITALS ARE ORDERED SUCH THAT THE ORBITALS THAT YOU WANT IN THE ACTIVE SPACE COME IN THE DESIRED ORDER**. In many cases this will require re-ordering and **CAREFUL INSPECTION** of the starting orbitals.

Attention

If you include orbitals in the active space that are nearly empty or nearly doubly occupied, convergence problems are likely. The SuperCI(PT) [430] and Newton-Raphson method are less prone to these problems.

3.13.6 Guess: Natural Orbitals Example

The initial guess and selection of the active orbitals is crucial for the success of any CASSCF calculation. Often, the default `!PModel` guess does not provide good starting orbitals or the correct active space. Which orbitals enter as active, depends on the system and requires the users insight. The ideal choice might vary with the molecules and calculation design. For instance, in the case of lanthanides/actinides, the fragment guess, is particularly successful and leads to a rapid convergence. A number of typical examples, including the fragment guess, can be found in the CASSCF tutorial. If the nature of the active space is *a priori* known, the *atomic valence space selection (AVAS)* offers a user-friendly guess alternative.

Irrespective of the origin of the initial starting orbitals, it is always recommended to inspect the orbitals prior the orbital optimization.

Tip

In many cases, natural orbitals of a simple correlated calculation of some kind provide an idea of which orbitals to include in a CASSCF calculation.

Let us illustrate this principle with a calculation on the Benzene molecule where we want to include all six π-orbitals in the active space. After doing a RHF calculation:

```
! RHF SV(P)

* int 0 1
C 0 0 0 0.000000 0.000 0.000
C 1 0 0 1.389437 0.000 0.000
C 2 1 0 1.389437 120.000 0.000
```

(continues on next page)

(continued from previous page)

```
C 3 2 1 1.389437 120.000 0.000
C 4 3 2 1.389437 120.000 0.000
C 5 4 3 1.389437 120.000 0.000
H 1 2 3 1.082921 120.000 180.000
H 2 1 3 1.082921 120.000 180.000
H 3 2 1 1.082921 120.000 180.000
H 4 3 2 1.082921 120.000 180.000
H 5 4 3 1.082921 120.000 180.000
H 6 5 4 1.082921 120.000 180.000
*
%Output
        Print[P_ReducedOrbPopMO_L]      1
End
```

We can look at the orbitals around the HOMO/LUMO gap:

```
                      12        13        14        15        16        17
                  -0.63810  -0.62613  -0.59153  -0.59153  -0.50570  -0.49833
                   2.00000   2.00000   2.00000   2.00000   2.00000   2.00000
                  --------  --------  --------  --------  --------  --------
 0 C  s               2.9       0.0       0.3       0.1       0.0       0.0
 0 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 0 C  px              1.4      12.4       5.9       0.3       0.0      11.2
 0 C  py              4.2       4.1      10.1       5.9       0.0       0.1
 0 C  dyz             0.0       0.0       0.0       0.0       0.1       0.0
 0 C  dx2y2           0.1       0.1       0.2       0.2       0.0       0.5
 0 C  dxy             0.4       0.0       0.0       0.2       0.0       0.0
 1 C  s               2.9       0.0       0.3       0.1       0.0       0.0
 1 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 1 C  px              1.4      12.4       5.9       0.3       0.0      11.2
 1 C  py              4.2       4.1      10.1       5.9       0.0       0.1
 1 C  dyz             0.0       0.0       0.0       0.0       0.1       0.0
 1 C  dx2y2           0.1       0.1       0.2       0.2       0.0       0.5
 1 C  dxy             0.4       0.0       0.0       0.2       0.0       0.0
 2 C  s               2.9       0.0       0.0       0.4       0.0       0.1
 2 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 2 C  px              5.7       0.0       0.0      20.9       0.0      10.1
 2 C  py              0.0      16.5       1.3       0.0       0.0       0.0
 2 C  dxz             0.0       0.0       0.0       0.0       0.1       0.0
 2 C  dx2y2           0.6       0.0       0.0       0.2       0.0       1.2
 2 C  dxy             0.0       0.1       0.5       0.0       0.0       0.0
 3 C  s               2.9       0.0       0.3       0.1       0.0       0.0
 3 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 3 C  px              1.4      12.4       5.9       0.3       0.0      11.2
 3 C  py              4.2       4.1      10.1       5.9       0.0       0.1
 3 C  dyz             0.0       0.0       0.0       0.0       0.1       0.0
 3 C  dx2y2           0.1       0.1       0.2       0.2       0.0       0.5
 3 C  dxy             0.4       0.0       0.0       0.2       0.0       0.0
 4 C  s               2.9       0.0       0.3       0.1       0.0       0.0
 4 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 4 C  px              1.4      12.4       5.9       0.3       0.0      11.2
 4 C  py              4.2       4.1      10.1       5.9       0.0       0.1
 4 C  dyz             0.0       0.0       0.0       0.0       0.1       0.0
 4 C  dx2y2           0.1       0.1       0.2       0.2       0.0       0.5
 4 C  dxy             0.4       0.0       0.0       0.2       0.0       0.0
 5 C  s               2.9       0.0       0.0       0.4       0.0       0.1
 5 C  pz              0.0       0.0       0.0       0.0      16.5       0.0
 5 C  px              5.7       0.0       0.0      20.9       0.0      10.1
 5 C  py              0.0      16.5       1.3       0.0       0.0       0.0
 5 C  dxz             0.0       0.0       0.0       0.0       0.1       0.0
 5 C  dx2y2           0.6       0.0       0.0       0.2       0.0       1.2
```

(continues on next page)

(continued from previous page)

```
 5 C  dxy            0.0       0.1       0.5       0.0       0.0       0.0
 6 H  s              7.5       0.0       7.5       2.5       0.0       2.5
 7 H  s              7.5       0.0       7.5       2.5       0.0       2.5
 8 H  s              7.5       0.0       0.0      10.0       0.0       9.9
 9 H  s              7.5       0.0       7.5       2.5       0.0       2.5
10 H  s              7.5       0.0       7.5       2.5       0.0       2.5
11 H  s              7.5       0.0       0.0      10.0       0.0       9.9

                    18        19        20        21        22        23
                -0.49833  -0.33937  -0.33937   0.13472   0.13472   0.18198
                 2.00000   2.00000   2.00000   0.00000   0.00000   0.00000
                --------  --------  --------  --------  --------  --------
 0 C  s              0.1       0.0       0.0       0.0       0.0       2.2
 0 C  pz             0.0       8.1      24.4       7.8      23.4       0.0
 0 C  px             0.1       0.0       0.0       0.0       0.0       0.6
 0 C  py            10.4       0.0       0.0       0.0       0.0       1.7
 0 C  dxz            0.0       0.4       0.2       0.7       0.7       0.0
 0 C  dyz            0.0       0.2       0.0       0.7       0.0       0.0
 0 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.2
 0 C  dxy            1.0       0.0       0.0       0.0       0.0       0.5
 1 C  s              0.1       0.0       0.0       0.0       0.0       2.2
 1 C  pz             0.0       8.1      24.4       7.8      23.4       0.0
 1 C  px             0.1       0.0       0.0       0.0       0.0       0.6
 1 C  py            10.4       0.0       0.0       0.0       0.0       1.7
 1 C  dxz            0.0       0.4       0.2       0.7       0.7       0.0
 1 C  dyz            0.0       0.2       0.0       0.7       0.0       0.0
 1 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.2
 1 C  dxy            1.0       0.0       0.0       0.0       0.0       0.5
 2 C  s              0.0       0.0       0.0       0.0       0.0       2.2
 2 C  pz             0.0      32.5       0.0      31.2       0.0       0.0
 2 C  px             0.0       0.0       0.0       0.0       0.0       2.2
 2 C  py            11.6       0.0       0.0       0.0       0.0       0.0
 2 C  dxz            0.0       0.1       0.0       0.3       0.0       0.0
 2 C  dyz            0.0       0.0       0.8       0.0       1.8       0.0
 2 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.7
 2 C  dxy            0.4       0.0       0.0       0.0       0.0       0.0
 3 C  s              0.1       0.0       0.0       0.0       0.0       2.2
 3 C  pz             0.0       8.1      24.4       7.8      23.4       0.0
 3 C  px             0.1       0.0       0.0       0.0       0.0       0.6
 3 C  py            10.4       0.0       0.0       0.0       0.0       1.7
 3 C  dxz            0.0       0.4       0.2       0.7       0.7       0.0
 3 C  dyz            0.0       0.2       0.0       0.7       0.0       0.0
 3 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.2
 3 C  dxy            1.0       0.0       0.0       0.0       0.0       0.5
 4 C  s              0.1       0.0       0.0       0.0       0.0       2.2
 4 C  pz             0.0       8.1      24.4       7.8      23.4       0.0
 4 C  px             0.1       0.0       0.0       0.0       0.0       0.6
 4 C  py            10.4       0.0       0.0       0.0       0.0       1.7
 4 C  dxz            0.0       0.4       0.2       0.7       0.7       0.0
 4 C  dyz            0.0       0.2       0.0       0.7       0.0       0.0
 4 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.2
 4 C  dxy            1.0       0.0       0.0       0.0       0.0       0.5
 5 C  s              0.0       0.0       0.0       0.0       0.0       2.2
 5 C  pz             0.0      32.5       0.0      31.2       0.0       0.0
 5 C  px             0.0       0.0       0.0       0.0       0.0       2.2
 5 C  py            11.6       0.0       0.0       0.0       0.0       0.0
 5 C  dxz            0.0       0.1       0.0       0.3       0.0       0.0
 5 C  dyz            0.0       0.0       0.8       0.0       1.8       0.0
 5 C  dx2y2          0.0       0.0       0.0       0.0       0.0       0.7
 5 C  dxy            0.4       0.0       0.0       0.0       0.0       0.0
 6 H  s              7.4       0.0       0.0       0.0       0.0      11.5
```

(continues on next page)

(continued from previous page)

```
 7 H   s                7.4        0.0        0.0        0.0        0.0       11.5
 8 H   s                0.0        0.0        0.0        0.0        0.0       11.5
 9 H   s                7.4        0.0        0.0        0.0        0.0       11.5
10 H   s                7.4        0.0        0.0        0.0        0.0       11.5
11 H   s                0.0        0.0        0.0        0.0        0.0       11.5
```

Since the molecule is aligned in the x-y plane, the desired orbitals must have 'pz' character. We see that the occupied π-orbitals number 16, 19, 20 and the unoccupied ones start with 21 and 22. However, the sixth high-lying π^*-orbital cannot easily be found. The task is is much easier using natural orbitals from a correlated calculation. Let us run a simple selected RI-MP2 calculation and look at the natural orbitals.

```
! RHF SV(P) RI-MP2 def2-svp/C
! moread
%moinp "Test-CASSCF-Benzene-1.gbw"

%mp2
 # TNat is an alias:
 #  - selected MP2 wave function (1e-5)
 #  - compute natural orbitals with the unrelaxed density.
 # alternative: "natorbs true" with  "density unrelaxed".
 # ==> produces a gbw file with the suffix ".mp2nat"
 tnat 1e-5
end

# ...etc, input of coordinates
```

The calculation prints the occupation numbers:

```
N[ 16] =   1.95798013
N[ 17] =   1.95798013
N[ 18] =   1.95330649
N[ 19] =   1.91852498
N[ 20] =   1.91852498
N[ 21] =   0.06544306
N[ 22] =   0.06544306
N[ 23] =   0.02820679
N[ 24] =   0.02405988
```

From these occupation number it becomes evident that there are several natural orbitals which are not quite doubly occupied MOs. The most promising orbitals are right at the HOMO-LOMO gap. These are typically the π and π^* orbitals. In a CASSCF(6,6), the orbitals 18-23 would be selected as active.

Tip

The first active orbital always starts with '(number of electrons - number of active electrons)/2'.

Let us see what these orbitals are before starting the full CASSCF optimization:

```
! SV(P)
! moread noiter
%moinp "Test-CASSCF-Benzene-2.mp2nat"

# Enable reduced orbital population per MO
%Output
Print[P_ReducedOrbPopMO_L] 1
End
...
```

Leading to:

```
                      ******************************
                      * LOEWDIN POPULATION ANALYSIS *
                      ******************************

      **** WARNING: LOEWDIN FINDS   41.4430078 ELECTRONS INSTEAD OF 42 ****
                      **** SKIPPING LOEWDIN ANALYSIS ****
-----------------------------------------
LOEWDIN REDUCED ORBITAL POPULATIONS PER MO
-----------------------------------------
THRESHOLD FOR PRINTING IS 0.1%%
...
                      18        19        20        21        22        23
                  -0.49831  -0.33935  -0.33935   0.13474   0.13474   0.18198
                   1.95331   1.91853   1.91853   0.06544   0.06544   0.02821
                  --------  --------  --------  --------  --------  --------
 0 C  pz             16.5       8.1      24.4       7.6      22.9      16.2
 0 C  dxz             0.0       0.4       0.2       1.4       0.7       0.1
 0 C  dyz             0.1       0.2       0.0       0.7       0.0       0.3
 1 C  pz             16.5       8.1      24.4       7.6      22.9      16.2
 1 C  dxz             0.0       0.4       0.2       1.4       0.7       0.1
 1 C  dyz             0.1       0.2       0.0       0.7       0.0       0.3
 2 C  pz             16.5      32.5       0.0      30.5       0.0      16.2
 2 C  dxz             0.1       0.1       0.0       0.1       0.0       0.4
 2 C  dyz             0.0       0.0       0.8       0.0       2.8       0.0
 3 C  pz             16.5       8.1      24.4       7.6      22.9      16.2
 3 C  dxz             0.0       0.4       0.2       1.4       0.7       0.1
 3 C  dyz             0.1       0.2       0.0       0.7       0.0       0.3
 4 C  pz             16.5       8.1      24.4       7.6      22.9      16.2
 4 C  dxz             0.0       0.4       0.2       1.4       0.7       0.1
 4 C  dyz             0.1       0.2       0.0       0.7       0.0       0.3
 5 C  pz             16.5      32.5       0.0      30.5       0.0      16.2
 5 C  dxz             0.1       0.1       0.0       0.1       0.0       0.4
 5 C  dyz             0.0       0.0       0.8       0.0       2.8       0.0
```

This shows us that these six orbitals are precisely the π/π^*-orbitals that we wanted to have active (you can also plot them to get even more insight). If the orbitals wouldn't be in the correct position to enter as active (18-23), orbitals can be swapped with the `rotate` keyword described *elsewhere*.

Now that we know that the desired orbitals are in the correct order, we can optimize the orbitals with CASSCF:

```
! SV(P)
! moread
%moinp "Test-CASSCF-Benzene-2.mrci.nat"

%casscf
  # define CAS(6,6)
  nel    6
  norb   6

  # ground state singlet
  mult   1
  nroots 1

  switchstep nr # for illustration purpose
end
```

To highlight the feature `SwitchStep` of the CASSCF program, we employ the Newton-Raphson method (NR) after a certain convergence has been reached (`SwitchStep NR` statement). In general, it is recommended to use the default convergence settings! The output of the CASSCF program is:

```
------------------
CAS-SCF ITERATIONS
```

(continues on next page)

(continued from previous page)

```
-----------------


MACRO-ITERATION   1:
--- Inactive Energy E0 = -224.09725414 Eh
CI-ITERATION   0:
-230.588253032   0.000000000000 (    0.00)
CI-PROBLEM SOLVED
DENSITIES MADE

<<<<<<<<<<<<<<<<<<INITIAL CI STATE CHECK>>>>>>>>>>>>>>>>>>

BLOCK  1 MULT= 1 NROOTS= 1
ROOT   0:  E=    -230.5882530315 Eh
0.89482 [     0]: 222000
0.02897 [    14]: 211110
0.01982 [    29]: 202020
0.01977 [     4]: 220200
0.01177 [    65]: 112011
0.01169 [    50]: 121101

<<<<<<<<<<<<<<<<<<INITIAL CI STATE CHECK>>>>>>>>>>>>>>>>>>

E(CAS)=  -230.588253032 Eh DE=    0.000000e+00
--- Energy gap subspaces: Ext-Act = 0.195   Act-Int = 0.127
--- current l-shift: Up(Ext-Act) = 1.40   Dn(Act-Int) = 1.47
N(occ)=  1.96393 1.90933 1.90956 0.09190 0.09208 0.03319
||g|| =     1.046979e-01 Max(G)=   -4.638985e-02 Rot=53,19
--- Orbital Update [SuperCI(PT)]
--- Canonicalize Internal Space
--- Canonicalize External Space
--- SX_PT (Skipped TA=0 IT=0): ||X|| =      0.063973050 Max(X)(83,23) =     -0.
↪035491133
--- SFit(Active Orbitals)

MACRO-ITERATION   2:
--- Inactive Energy E0 = -224.09299157 Eh
CI-ITERATION   0:
-230.590141151   0.000000000000 (    0.00)
CI-PROBLEM SOLVED
DENSITIES MADE
E(CAS)=  -230.590141151 Eh DE=   -1.888119e-03
--- Energy gap subspaces: Ext-Act = 0.202   Act-Int = 0.126
--- current l-shift: Up(Ext-Act) = 0.90   Dn(Act-Int) = 0.97
N(occ)=  1.96182 1.90357 1.90364 0.09771 0.09777 0.03549
||g|| =     2.971340e-02 Max(G)=   -8.643429e-03 Rot=52,20
--- Orbital Update [SuperCI(PT)]
--- Canonicalize Internal Space
--- Canonicalize External Space
--- SX_PT (Skipped TA=0 IT=0): ||X|| =      0.009811159 Max(X)(67,21) =     -0.
↪003665750
--- SFit(Active Orbitals)

MACRO-ITERATION   3:
===>>> Convergence to 3.0e-02 achieved - switching to Step=NR
--- Inactive Energy E0 = -224.07872151 Eh
CI-ITERATION   0:
-230.590260496   0.000000000000 (    0.00)
CI-PROBLEM SOLVED
DENSITIES MADE
E(CAS)=  -230.590260496 Eh DE=   -1.193453e-04
```

(continues on next page)

(continued from previous page)

```
--- Energy gap subspaces: Ext-Act = 0.203   Act-Int = 0.125
--- current l-shift: Up(Ext-Act) = 0.73   Dn(Act-Int) = 0.81
N(occ)=  1.96145 1.90275 1.90278 0.09856 0.09857 0.03589
||g|| =     8.761362e-03 Max(G)=    4.388664e-03 Rot=43,19
--- Orbital Update [        NR]
AUGHESS-ITER   0: E=    -0.000016434 <r|r>= 2.70127912e-05
AUGHESS-ITER   1: E=    -0.000021148 <r|r>= 2.91399830e-06
AUGHESS-ITER   2: E=    -0.000021780 <r|r>= 4.01336069e-07 => CONVERGED
DE(predicted)=   -0.000010890 First Element= 0.999987718
<X(rot)|X(rot)>=      0.000024564
--- SFit(Active Orbitals)

MACRO-ITERATION   4:
--- Inactive Energy E0 = -224.07787812 Eh
CI-ITERATION   0:
-230.590271490   0.000000000000 (    0.00)
CI-PROBLEM SOLVED
DENSITIES MADE
E(CAS)=  -230.590271490 Eh DE=   -1.099363e-05
--- Energy gap subspaces: Ext-Act = 0.202   Act-Int = 0.125
--- current l-shift: Up(Ext-Act) = 0.40   Dn(Act-Int) = 0.47
N(occ)=  1.96135 1.90267 1.90267 0.09866 0.09866 0.03599
||g|| =     6.216730e-04 Max(G)=    1.417079e-04 Rot=66,13
---- THE CAS-SCF GRADIENT HAS CONVERGED ----
--- FINALIZING ORBITALS ---
---- DOING ONE FINAL ITERATION FOR PRINTING ----
--- Forming Natural Orbitals
--- Canonicalize Internal Space
--- Canonicalize External Space

MACRO-ITERATION   5:
--- Inactive Energy E0 = -224.07787811 Eh
--- All densities will be recomputed
CI-ITERATION   0:
-230.590271485   0.000000000000 (    0.00)
CI-PROBLEM SOLVED
DENSITIES MADE
E(CAS)=  -230.590271485 Eh DE=    5.179942e-09
--- Energy gap subspaces: Ext-Act = -0.242   Act-Int = -0.002
--- current l-shift: Up(Ext-Act) = 0.84   Dn(Act-Int) = 0.60
N(occ)=  1.96135 1.90267 1.90267 0.09866 0.09866 0.03599
||g|| =     6.216710e-04 Max(G)=    1.544017e-04 Rot=29,12
--------------
CASSCF RESULTS
--------------

Final CASSCF energy       : -230.590271485 Eh   -6274.6803 eV
```

First of all, we see how the program cycles between CI-vector optimization and orbital optimization steps (so-called unfolded two-step procedure). After 3 iterations, the program switches to the Newton-Raphson solver which then converges very rapidly. Orbital optimization with the Newton-Raphson solver is limited to smaller sized molecules, as the program produces lengthy integrals and Hessian files. In the majority of situations the default converger (SuperCI(PT)) is the preferred choice.[430] In difficult situations, one should consider the *TRAH optimizer*.

3.13.7 Guess: Atomic Valence Active Space (AVAS)

Very good starting orbitals that are targeted to a specific user-given active space can be generated with the Atomic Valence Active Space (AVAS) procedure. [449, 450] The general idea is that the user provides a set of atomic orbitals (AO) of a minimal basis set that are sufficient to qualitatively represent the final CASSCF active orbitals. Typical examples are

- p_z orbitals of a π system chromophore in a molecule
- five valence (or 10 double-shell) d orbitals of a transition-metal (TM) atom in a molecule
- seven valence (or 14 double-shell) f orbitals of a lanthanide or actinide atom in a molecule

Then, by the help of linear algebra (singular-value decomposition) AVAS rotates the starting molecular orbitals (MOs) such that they have maximum overlap with the target AOs. With those rotated MOs that have a sufficiently large singular value (> 0.4 (default)) are considered as active orbitals. In that manner, AVAS automatically determines an active space, i.e. the number of active orbitals and electrons, that is now specified by the target AOs.

Warning

AVAS overwrites the number of active orbitals and electrons in the `%casscf` block!

As a first example, we now consider $CuCl_4^-$ in a minimal active space

```
! cc-pvtz TightSCF Def2/JK PModel NoIter
! AVAS(Valence-D)

%maxcore 3000

%paras
 cucl  = 2.291
end

* int -1 1
Cu  0  0  0   0.0     0.0     0.0
Cl   1  0  0   {cucl}  0.0      0.0
Cl   1  2  0   {cucl}  90.0     0.0
Cl   1  3  2   {cucl}  90.0   180.
Cl   1  4  3   {cucl}  90.0   180.
*
```

The keyword `! AVAS(Valence-D)` seeks for all transition-metal atoms in the molecule and inserts a single minimal d basis function for each TM atom. All five component M_L of the basis function are then considered. The AVAS procedure prints singular / eigen values for the occupied and virtual orbital space and easily finds the desired minimal active space CAS(9,5).

```
-------------------------------------------------
INITIAL GUESS: Atomic Valence Active space (AVAS)
-------------------------------------------------

AVAS threshold         : 0.400000
AVAS minimal basis set : MINAO
AVAS list              : Shell | 3 2  0> at atom 0 (system 0)
 \\\\\: Shell | 3 2  1> at atom 0 (system 0)
 \\\\\: Shell | 3 2 -1> at atom 0 (system 0)
 \\\\\: Shell | 3 2  2> at atom 0 (system 0)
 \\\\\: Shell | 3 2 -2> at atom 0 (system 0)

AVAS P matrix eig. val (  Occupied) : 0.966698
AVAS P matrix eig. val (  Occupied) : 0.974913
AVAS P matrix eig. val (  Occupied) : 0.977443
```

(continues on next page)

(continued from previous page)

```
AVAS P matrix eig. val (  Occupied) : 0.977443
AVAS P matrix eig. val (  Occupied) : 0.985233
AVAS P matrix eig. val (   Virtual) : (0.032829)
AVAS P matrix eig. val (   Virtual) : (0.024687)
AVAS P matrix eig. val (   Virtual) : (0.022047)
AVAS P matrix eig. val (   Virtual) : (0.022047)
AVAS P matrix eig. val (   Virtual) : (0.014546)

AVAS electrons          : 9
AVAS orbitals           : 5
```

The five initial active orbitals after being processed by AVAS indeed look like the desired Cu d-orbitals.

(a) $\sigma^o{}_1 = 0.977$ (b) $\sigma^o{}_3 = 0.975$ (c) $\sigma^o{}_0 = 0.985$

(d) $\sigma^o{}_4 = 0.967$ (e) $\sigma^o{}_2 = 0.977$

Fig. 3.14: Initial Minimal AS orbitals of $CuCl_4^-$ generated by AVAS.

The same calculation can be started also by using the `%scf avas ... end end` block.

```
%scf
 avas
  system
   shell   3,  3,  3,  3,  3
   l       2,  2,  2,  2,  2
   m_l     0,  1, -1,  2, -2
   center  0,  0, 0,  0, 0
  end
 end
end
```

Here, it is also possible to use target basis functions at different atoms (`center`) and to select only a subset of

functions in a shell (m_l). Note that if not all functions of a shell (3p, 5d, 7f) are selected, the molecule should be oriented manually to accomplish the desired basis function overlap.

AVAS can be also used very conveniently in the same fashion for double-d shell calculations with transition-metal complexes (`! AVAS(Double-D)`). For each 3d transition-metal center in a molecule all 3d and 4d target functions are considered. Similarly, double-shell active spaces can be also set up for 4d and 5d transition-metal complexes.

There is also a similar keyword for lanthanides and actinides. `! AVAS( Valence-F )` attempts to set up an active space with 7 f functions for each lanthanide or actinide atom in a molecule. There is also the possibility to run double-f shell calculations using the `! AVAS( Double-F )` keyword.

To avoid this issue for π active space calculation, all three 2p target AOs are considered first but they are weighted by the three component of the principle axis of inertia with the largest moment. [450] For those inertia moment calculations, masses are ignored and only the centers of the desired target p AO are considered.

For a CAS(10,9) π-active space calculation on tryptophan, the AVAS input read

```
! cc-pVTZ PModel NoIter

%scf
 avas
  tol 0.4
  system
   center 0, 1, 2, 3, 4, 5, 10, 11, 12
   type  pz, pz, pz, pz, pz, pz, pz, pz, pz
  end
 end
end

* xyz 0 1
 C          0.4512549872        2.3796411953        0.0577773122
 C          0.1094760583        1.0035547288       -0.1566676092
 C          1.7801675822        2.8072137170        0.2571892289
 C          2.7806901872        1.8262977582        0.2356692574
 C          2.4656511421        0.4546661378        0.0230301052
 C          1.1452272475        0.0339609344       -0.1736410480
 H          2.0259509453        3.8615102004        0.4187388370
 H          3.8237760997        2.1214062744        0.3833595222
 H          3.2752609035       -0.2812140701        0.0109117373
 H          0.9203743659       -1.0244858148       -0.3388820373
 C         -1.3215206968        0.9316755285       -0.3177965838
 C         -1.7965156128        2.2386249398       -0.2022300378
 N         -0.7296902726        3.0958334808        0.0227806512
 H         -0.8107596679        4.0971334562        0.1485860796
 H         -2.8167763088        2.6080109542       -0.2688439980
 C         -2.1029028025       -0.3291635000       -0.5688909937
 C         -3.4238543678       -0.4065989881        0.2267575199
 H         -1.4745479852       -1.1909157350       -0.2884954137
 H         -2.3461010681       -0.4478113906       -1.6421457333
 C         -3.9423325138       -1.8379287141        0.1258142785
 N         -4.3742952299        0.5836598444       -0.2812451173
 H         -3.2051519657       -0.1892488262        1.2846794690
 O         -3.2924970778       -2.6708957465        0.9924074621
 O         -4.8043368378       -2.2232843366       -0.6488988164
 H         -3.6480373076       -3.5631013900        0.8277551234
 H         -5.2270970579        0.5578136152        0.2816027849
 H         -4.6658127461        0.2911757460       -1.2180819802
*
```

and leads to the following output

```
-------------------------------------------------
INITIAL GUESS: Atomic Valence Active space (AVAS)
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------------

   AVAS threshold         : 0.400000
   AVAS minimal basis set : AUTO
   AVAS list              : Shell | 2 1  0> at atom 0 (system 0, type pz)
                          : Shell | 2 1  1> at atom 0 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 0 (system 0, type pz)
                          : Shell | 2 1  0> at atom 1 (system 0, type pz)
                          : Shell | 2 1  1> at atom 1 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 1 (system 0, type pz)
                          : Shell | 2 1  0> at atom 2 (system 0, type pz)
                          : Shell | 2 1  1> at atom 2 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 2 (system 0, type pz)
                          : Shell | 2 1  0> at atom 3 (system 0, type pz)
                          : Shell | 2 1  1> at atom 3 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 3 (system 0, type pz)
                          : Shell | 2 1  0> at atom 4 (system 0, type pz)
                          : Shell | 2 1  1> at atom 4 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 4 (system 0, type pz)
                          : Shell | 2 1  0> at atom 5 (system 0, type pz)
                          : Shell | 2 1  1> at atom 5 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 5 (system 0, type pz)
                          : Shell | 2 1  0> at atom 10 (system 0, type pz)
                          : Shell | 2 1  1> at atom 10 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 10 (system 0, type pz)
                          : Shell | 2 1  0> at atom 11 (system 0, type pz)
                          : Shell | 2 1  1> at atom 11 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 11 (system 0, type pz)
                          : Shell | 2 1  0> at atom 12 (system 0, type pz)
                          : Shell | 2 1  1> at atom 12 (system 0, type pz)
                          : Shell | 2 1 -1> at atom 12 (system 0, type pz)

   AVAS P matrix eig. val (  Occupied) : (0.000004)
   AVAS P matrix eig. val (  Occupied) : (0.000014)
   AVAS P matrix eig. val (  Occupied) : (0.000292)
   AVAS P matrix eig. val (  Occupied) : (0.040014)
   AVAS P matrix eig. val (  Occupied) : 0.978162
   AVAS P matrix eig. val (  Occupied) : 0.986637
   AVAS P matrix eig. val (  Occupied) : 0.993225
   AVAS P matrix eig. val (  Occupied) : 0.994300
   AVAS P matrix eig. val (  Occupied) : 0.996447
   AVAS P matrix eig. val (   Virtual) : 0.999996
   AVAS P matrix eig. val (   Virtual) : 0.999986
   AVAS P matrix eig. val (   Virtual) : 0.999708
   AVAS P matrix eig. val (   Virtual) : 0.959986
   AVAS P matrix eig. val (   Virtual) : (0.021838)
   AVAS P matrix eig. val (   Virtual) : (0.013363)
   AVAS P matrix eig. val (   Virtual) : (0.006775)
   AVAS P matrix eig. val (   Virtual) : (0.005700)
   AVAS P matrix eig. val (   Virtual) : (0.003553)

   AVAS electrons         : 10
   AVAS orbitals          : 9

                       ------------------
                       INITIAL GUESS DONE (   0.3 sec)
                       ------------------
```

and initial active orbitals.

Fig. 3.15: Initial π AS orbitals of tryptophan generated by AVAS.

It is also possible to specify the number of active electrons `nel` and orbitals `norb` directly. For such a calculation, the AVAS singular value decomposition threshold `tol` is ignored. In the following calculation, the strongly occupied orbital from the previous CAS(10,9) (σ^o_4 in Fig. 3.15) calculation is omitted.

```
%scf
 avas
  system
   norb 8
   nel 8
   center 0, 1, 2, 3, 4, 5, 10, 11, 12
   type  pz, pz, pz, pz, pz, pz, pz, pz, pz
  end
 end
end
```

It is also possible to do the AVAS start MO generation for several `systems` independently and then re-orthonormalize all MOs at the end similar to [450]. This becomes interesting for generating starting orbitals for multiple π chromophores like the bridged bithiophene biradical

```
%scf
 avas
  system
   norb 4
   nel  3
   center   0, 1, 2, 3, 4 # C / S atoms system 1
   type    pz, pz, pz, pz, pz
  end
  system
```

(continues on next page)

(continued from previous page)

```
   norb 4
   nel  5
   center  38, 39, 40, 41, 43 # C / S atoms system 2
   type    pz, pz, pz, pz, pz
  end
 end
end
```

or the FeTPP molecule.

```
! SVP NoIter
! PModel

%scf
 avas
  system
   center 0
   type d
  end
  system
   center  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,␣
↪19, 20, 29, 30, 31, 32
   type    pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz, pz,␣
↪pz, pz, pz, pz, pz, pz
  end
 end
end

*xyz 0 1
Fe 0.0000 0.0000  0.0000
N  1.9764 0.0000  0.0000
N  0.0000 0.0000  1.9884
N -1.9764 0.0000  0.0000
N  0.0000 0.0000 -1.9884
C  2.8182 0.0000 -1.0903
C  2.8182 0.0000  1.0903
C  1.0918 0.0000  2.8249
C -1.0918 0.0000  2.8249
C -2.8182 0.0000  1.0903
C -2.8182 0.0000 -1.0903
C -1.0918 0.0000 -2.8249
C  1.0918 0.0000 -2.8249
C  4.1961 0.0000 -0.6773
C  4.1961 0.0000  0.6773
C  0.6825 0.0000  4.1912
C -0.6825 0.0000  4.1912
C -4.1961 0.0000  0.6773
C -4.1961 0.0000 -0.6773
C -0.6825 0.0000 -4.1912
C  0.6825 0.0000 -4.1912
H  5.0441 0.0000 -1.3538
H  5.0441 0.0000  1.3538
H  1.3558 0.0000  5.0416
H -1.3558 0.0000  5.0416
H -5.0441 0.0000  1.3538
H -5.0441 0.0000 -1.3538
H -1.3558 0.0000 -5.0416
H  1.3558 0.0000 -5.0416
C  2.4150 0.0000  2.4083
C -2.4150 0.0000  2.4083
C -2.4150 0.0000 -2.4083
```

(continues on next page)

(continued from previous page)

```
C  2.4150 0.0000 -2.4083
H  3.1855 0.0000  3.1752
H -3.1855 0.0000  3.1752
H -3.1855 0.0000 -3.1752
H  3.1855 0.0000 -3.1752
*
```

For those systems, all AVAS starting MOs have the desired π or d character as illustrated for the "active frontier orbitals" in Fig. 3.16.

(a) AVAS HOMO

(b) AVAS LUMO

(c) AVAS HOMO

(d) AVAS LUMO

Fig. 3.16: Initial HOMO and LUMO AVAS orbitals of a bridged bithiophene biradical and FeTPP.

3.13.8 Example: Symmetry

The CASSCF program can make some use of symmetry. Thus, it is possible to do the CI calculations separated by irreducible representations. This allows one to calculate electronic states in a more controlled fashion.

Let us look at a simple example: C_2H_4. We first generate symmetry adapted MP2 natural orbitals. Since we opt for initial guess orbitals, the computationally cheaper unrelaxed density suffices:

```
! def2-TZVP def2-TZVP/C UseSym RI-MP2 conv # conventional is faster for small␣
↪molecules

%mp2
density unrelaxed  # unrelaxed is sufficient for guess orbitals.
natorbs true
end

* int 0 1
C 0 0 0  0 0 0
C 1 0 0  1.35 0 0
H 1 2 0  1.1 120 0
H 1 2 3  1.1 120 180
H 2 1 3  1.1 120 0
H 2 1 3  1.1 120 180
*
```

The program does the following. It first identifies the group correctly as D_{2h} and sets up its irreducible representations. The process detects symmetry within `SymThresh` (10^{-4}) and purifies the geometry thereafter:

```
------------------
SYMMETRY DETECTION
------------------
The point group will now be determined using a tolerance of 1.0000e-04.
Splitting atom subsets according to nuclear charge, mass and basis set.
Splitting atom subsets according to distance from the molecule's center.
Identifying relative distance patterns of the atoms.
Splitting atom subsets according to atoms' relative distance patterns.
Bring atoms of each subset into input order.
The molecule is planar.
The molecule has a center of inversion.
Analyzing the first atom subset for its symmetry.
The atoms in the selected subset form a 4-gon with alternating side lengths.
Testing point group D2h.
Success!
This point group has been found:    D2h
Largest non-degenerate subgroup:    D2h


Symmetry-perfected Cartesians (point group D2h):

Atom          Symmetry-perfected Cartesians (x, y, z; au)
  0      -1.275565140397    0.000000000000    0.000000000000
  1       1.275565140397    0.000000000000    0.000000000000
  2      -2.314914514054    1.800205921988    0.000000000000
  3      -2.314914514054   -1.800205921988    0.000000000000
  4       2.314914514054    1.800205921988    0.000000000000
  5       2.314914514054   -1.800205921988    0.000000000000


-------------------------------------------------
SYMMETRY-PERFECTED CARTESIAN COORDINATES (A.U.)
-------------------------------------------------
Warning (ORCA_SYM): Coordinates were not cleaned so far!
```

(continues on next page)

(continued from previous page)

```
------------------
SYMMETRY REDUCTION
------------------
ORCA supports only abelian point groups.
It is now checked, if the determined point group is supported:
Point Group ( D2h   ) is          ... supported

(Re)building abelian point group:
Creating Character Table          ... done
Making direct product table       ... done
Constructing symmetry operations  ... done
Creating atom transfer table      ... done
Creating asymmetric unit          ... done

----------------------
ASYMMETRIC UNIT IN D2h
----------------------
  #  AT     MASS              COORDS (A.U.)             BAS
   0 C   12.0110  -1.27556514   0.00000000   0.00000000   0
   2 H    1.0080  -2.31491451   1.80020592   0.00000000   0

----------------------
SYMMETRY ADAPTED BASIS
----------------------
The coefficients for the symmetry adapted linear combinations (SALCS)
of basis functions will now be computed:
Number of basis functions         ...    86
Preparing memory                  ... done
Constructing Gamma(red)           ... done
Reducing Gamma(red)               ... done
Constructing SALCs                ... done
Checking SALC integrity           ... nothing suspicious
Normalizing SALCs                 ... done

Storing the symmetry object:
Symmetry file                     ... Test-SYM-CAS-C2H4-1.sym.tmp
Writing symmetry information      ... done
```

It then performs the SCF calculation and keeps the symmetry in the molecular orbitals.

```
NO   OCC          E(Eh)            E(eV)    Irrep
   0   2.0000     -11.236728      -305.7669    1-Ag
   1   2.0000     -11.235157      -305.7242    1-B3u
   2   2.0000      -1.027144       -27.9500    2-Ag
   3   2.0000      -0.784021       -21.3343    2-B3u
   4   2.0000      -0.641566       -17.4579    1-B2u
   5   2.0000      -0.575842       -15.6694    3-Ag
   6   2.0000      -0.508313       -13.8319    1-B1g
   7   2.0000      -0.373406       -10.1609    1-B1u
   8   0.0000       0.139580         3.7982    1-B2g
   9   0.0000       0.171982         4.6799    4-Ag
  10   0.0000       0.195186         5.3113    3-B3u
  11   0.0000       0.196786         5.3548    2-B2u
  12   0.0000       0.242832         6.6078    2-B1g
  13   0.0000       0.300191         8.1686    5-Ag
  14   0.0000       0.326339         8.8801    4-B3u
... etc
```

The MP2 module does not take any advantage of this information but produces natural orbitals that are symmetry adapted:

```
N[  0](B3u) =   2.00000360
N[  1]( Ag) =   2.00000219
N[  2]( Ag) =   1.98056435
N[  3](B3u) =   1.97195041
N[  4](B2u) =   1.96746753
N[  5](B1g) =   1.96578954
N[  6]( Ag) =   1.95864726
N[  7](B1u) =   1.93107098
N[  8](B2g) =   0.04702701
N[  9](B3u) =   0.02071784
N[ 10](B2u) =   0.01727252
N[ 11]( Ag) =   0.01651489
N[ 12](B1g) =   0.01602695
N[ 13](B3u) =   0.01443373
N[ 14](B1u) =   0.01164204
N[ 15]( Ag) =   0.01008617
N[ 16](B2u) =   0.00999302
N[ 17]( Ag) =   0.00840326
N[ 18](B3g) =   0.00795053
N[ 19](B3u) =   0.00532044
N[ 20]( Au) =   0.00450556
etc.
```

From this information and visual inspection you will know what orbitals you will have in the active space:

These natural orbitals can then be fed into the CASSCF calculation. We perform a simple calculation in which we keep the ground state singlet (A_{1g} symmetry, irrep=0) and the first excited triplet state (B_{3u} symmetry, irrep=7). In general the ordering of irreps follows standard conventions and in case of doubt you will find the relevant number for each irrep in the output.

For example, here (using `!LargePrint`):

```
----------------------------
CHARACTER TABLE OF GROUP D2h
----------------------------
GAMMA  O1   O2   O3   O4   O5   O6   O7   O8
 Ag :  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
 B1g:  1.0  1.0 -1.0 -1.0  1.0  1.0 -1.0 -1.0
 B2g:  1.0 -1.0  1.0 -1.0  1.0 -1.0  1.0 -1.0
 B3g:  1.0 -1.0 -1.0  1.0  1.0 -1.0 -1.0  1.0
 Au :  1.0  1.0  1.0  1.0 -1.0 -1.0 -1.0 -1.0
 B1u:  1.0  1.0 -1.0 -1.0 -1.0 -1.0  1.0  1.0
 B2u:  1.0 -1.0  1.0 -1.0 -1.0  1.0 -1.0  1.0
 B3u:  1.0 -1.0 -1.0  1.0 -1.0  1.0  1.0 -1.0

---------------------------------
DIRECT PRODUCT TABLE OF GROUP D2h
---------------------------------
   **   Ag  B1g B2g B3g Au  B1u B2u B3u

 Ag     Ag  B1g B2g B3g Au  B1u B2u B3u
 B1g    B1g Ag  B3g B2g B1u Au  B3u B2u
 B2g    B2g B3g Ag  B1g B2u B3u Au  B1u
 B3g    B3g B2g B1g Ag  B3u B2u B1u Au
 Au     Au  B1u B2u B3u Ag  B1g B2g B3g
 B1u    B1u Au  B3u B2u B1g Ag  B3g B2g
 B2u    B2u B3u Au  B1u B2g B3g Ag  B1g
 B3u    B3u B2u B1u Au  B3g B2g B1g Ag
```

We use the following input for CASSCF, where we tightened the integral cut-offs and the convergence criteria using `!VeryTightSCF`.

```
! def2-TZVP  Conv UseSym
! moread
%moinp "Test-SYM-CAS-C2H4-1.mp2nat"
%casscf
  # define CAS(4,4)
  nel            4
  norb           4

  # This is only here to show that NR can also be used from
  # the start with orbstep
  orbstep    nr
  switchstep nr

  # The lowest singet and triplet states. The new feature
  # is the array "irrep" that lets you give the irrep for
  # a given block. Thus, now you can have several blocks of
  # the same multiplicity but different spatial symmetry
  irrep       0,7
  mult        1,3
  nroots      1,1
end

* int 0 1
C 0 0 0  0 0 0
C 1 0 0  1.35 0 0
H 1 2 0  1.1 120 0
H 1 2 3  1.1 120 180
H 2 1 3  1.1 120 0
H 2 1 3  1.1 120 180
*
```

And gives:

```
------------
SCF SETTINGS
------------
Hamiltonian:
 Ab initio Hamiltonian  Method          .... Hartree-Fock(GTOs)


General Settings:
 Integral files         IntName         .... Test-SYM-CAS-C2H4-1
 Hartree-Fock type      HFTyp           .... CASSCF
 Total Charge           Charge          ....    0
 Multiplicity           Mult            ....    1
 Number of Electrons    NEL             ....   16
 Basis Dimension        Dim             ....   86
 Nuclear Repulsion      ENuc            ....     32.9609050695 Eh

 Symmetry handling      UseSym         .... ON
 Point group                           .... D2h
 Used point group                      .... D2h
 Number of irreps                      .... 8
   Irrep   Ag has   19 symmetry adapted basis functions (ofs=   0)
   Irrep  B1g has   12 symmetry adapted basis functions (ofs=  19)
   Irrep  B2g has    8 symmetry adapted basis functions (ofs=  31)
   Irrep  B3g has    4 symmetry adapted basis functions (ofs=  39)
   Irrep   Au has    4 symmetry adapted basis functions (ofs=  43)
   Irrep  B1u has    8 symmetry adapted basis functions (ofs=  47)
   Irrep  B2u has   12 symmetry adapted basis functions (ofs=  55)
   Irrep  B3u has   19 symmetry adapted basis functions (ofs=  67)
```

And further in the CASSCF program:

```
Symmetry handling      UseSym     ... ON
Point group                       ... D2h
Used point group                  ... D2h
Number of irreps                  ... 8
   Irrep   Ag has   19 SALCs (ofs=   0) #(closed)=   2 #(active)=   1
   Irrep  B1g has   12 SALCs (ofs=  19) #(closed)=   1 #(active)=   0
   Irrep  B2g has    8 SALCs (ofs=  31) #(closed)=   0 #(active)=   1
   Irrep  B3g has    4 SALCs (ofs=  39) #(closed)=   0 #(active)=   0
   Irrep   Au has    4 SALCs (ofs=  43) #(closed)=   0 #(active)=   0
   Irrep  B1u has    8 SALCs (ofs=  47) #(closed)=   0 #(active)=   1
   Irrep  B2u has   12 SALCs (ofs=  55) #(closed)=   1 #(active)=   0
   Irrep  B3u has   19 SALCs (ofs=  67) #(closed)=   2 #(active)=   1
 Symmetries of active orbitals:
   MO =    6  IRREP= 0 (Ag)
   MO =    7  IRREP= 5 (B1u)
   MO =    8  IRREP= 2 (B2g)
   MO =    9  IRREP= 7 (B3u)

Setting up the integral package       ... done
Building the CAS space                ... done (7 configurations for Mult=1␣
↪Irrep=0)
Building the CAS space                ... done (4 configurations for Mult=3␣
↪Irrep=7)
```

Note that the irrep occupations and active space irreps will be frozen to what they are upon entering the CASSCF program. This helps to setup the CI problem.

After which it smoothly converges to give:

```
    6:  1.986258       -0.753012    -20.4905   3-Ag
    7:  1.457849       -0.291201     -7.9240   1-B1u
    8:  0.541977        0.100890      2.7454   1-B2g
    9:  0.013915        0.964186     26.2368   3-B3u
```

As well as:

```
-----------------------------
SA-CASSCF TRANSITION ENERGIES
-----------------------------

LOWEST ROOT =    -78.110314788 Eh -2125.490 eV

STATE   ROOT MULT IRREP DE/a.u.    DE/eV    DE/cm**-1
   1:    0    3  B3u   0.163741     4.456  35937.1
```

3.13.9 Example: Breaking Chemical Bonds

Let us turn to the breaking of chemical bonds. As a first example we study the dissociation of the H_2 molecule. Scanning a bond, we have two potential setups for the calculation: a) scan from the inside to the outside or b) from the outside to inside. Of course both setups yield identical results, but they differ in practical aspects i.e. convergence properties. In general, **scanning from the outside to the inside is the recommended procedure**. Using the default guess (PModel), starting orbitals are much easier identified than at shorter distances, where the antibonding orbitals are probably 'impure' and hence would require some additional preparation. To ensure a smooth potential energy surface, in all subsequent geometry steps, ORCA reads the converged CASSCF orbitals from the previous geometry step. In the following, `TightSCF` is used to tighten the convergence settings of CASSCF.

```
!Def2-SVP TightSCF

%casscf
 # define CAS(2,2)
```

(continues on next page)

(continued from previous page)

```
 nel         2
 norb        2
 # singlet ground state
 mult        1
 nroots      1
end

# Scanning from the outside to the inside
%paras
 R [4.1 3.8 3.5 3.2 2.9 2.6 2.4 2.2
    2 1.7 1.5 1.3 1.1 1 0.9 0.8
    0.75 0.7 0.65 0.6]
end

* xyz 0 1
 H   0.0   0.0   0.0
 H   0.0   0.0   {R}
end
```

The resulting potential energy surface (PES) is depicted in Fig. 3.17 together with PESs obtained from RHF and broken-symmetry UHF calculations (input below).

```
! RHF Def2-SVP TightSCF

# etc...
```

And

```
! UHF Def2-SVP TightSCF

%scf
  FlipSpin 1
  FinalMs 0.0
end

# etc...
```

Note

The `FlipSpin` option does not work together with the parameter scan. Only the first structure will undergo a spin flip. Therefore, at the current status, a separate input file (including the coordinates or with a corresponding coordinate file) has to be provided for each structure that is scanned along the PES.

Fig. 3.17: Potential Energy Surface of the H_2 molecule from RHF, UHF and CASSCF(2,2) calculations (Def2-SVP basis).

It is obvious, that the CASSCF surface is concise and yields the correct dissociation behavior. The RHF surface is roughly parallel to the CASSCF surface in the vicinity of the minimum but then starts to fail badly as the H-H bond starts to break. The broken-symmetry UHF solution is identical to RHF in the vicinity of the minimum and dissociates correctly. It is, however, of rather mediocre quality in the intermediate region where it follows the RHF surface.

A more challenging case is to dissociate the N-N bond of the N_2 molecule correctly. Using CASSCF with the six p-orbitals we get a nice potential energy curve (The depth of the minimum is still too shallow compared to experiment by some 1 eV or so. A good dissociation energy requires a dynamic correlation treatment on top of CASSCF and a larger basis set).

Fig. 3.18: Potential Energy Surface of the N_2 molecule from CASSCF(6,6) calculations (Def2-SVP basis).

One can use the H_2 example to illustrate the state-averaging feature. Since we have two active electrons we have two singlets and one triplet. Let us average the orbitals over these three states (we take equal weights for all multiplicity blocks):

```
!Def2-SVP TightSCF

%casscf
  # define CAS(2,2)
  nel         2
  norb        2

  # state-averaged triplet and singlet states
  mult        3,1
  nroots      1,2
end

# Scanning from the outside to the inside
%paras
R [4.1 3.8 3.5 3.2 2.9 2.6 2.4 2.2
   2 1.7 1.5 1.3 1.1 1 0.9 0.8
   0.75 0.7 0.65 0.6]
end

* xyz 0 1
 H 0 0 0
 H 0 0 {R}
end
```

which gives:

Fig. 3.19: State averaged CASSCF(2,2) calculations on H_2 (two singlets, one triplet; Def2-SVP basis). The grey curve is the ground state CASSCF(2,2) curve

One observes, that the singlet and triplet ground states become degenerate for large distances (as required) while the second singlet becomes the ionic singlet state which is high in energy. If one compares the lowest root of the state-averaged calculation (in green) with the dedicated ground state calculation (in gray) one gets an idea of the energetic penalty that is associated with averaged as opposed to dedicated orbitals.

A more involved example is the rotation around the double bond in C_2H_4. Here, the π-bond is broken as one twists the molecule. The means the proper active space consists of two active electron in two orbitals.

The input is (for fun, we average over the lowest two singlets and the triplet):

```
!def2-SVP TightSCF

%casscf
 # define CAS(2,2)
 nel        2
 norb       2
 # state-average triplet and singlet states
 mult       3,1
 nroots     1,2
end

%paras
 Alpha = 0,180,37
end

* int 0 1
 C  0  0  0  0      0   0
 C  1  0  0 1.34    0   0
 H  1  2  0 1.07  120   0
 H  1  2  3 1.07  120 180
 H  2  1  3 1.07  120 {Alpha}
```

(continues on next page)

(continued from previous page)

```
 H  2  1  3 1.07  120 {Alpha+180}
edn
```

Fig. 3.20: State averaged CASSCF(2,2) calculations on C_2H_4 (two singlets, one triplet; SV(P) basis). The grey curve is the state averaged energy.

We can see from this plot, that the CASSCF method produces a nice ground state surface with the correct periodicity and degeneracy at the end points, which represent the planar ethylene molecule. At 90° one has a weakly coupled diradical and the singlet and triplet states become nearly degenerate, again as expected. Calculations with larger basis sets and inclusion of dynamic correlation would give nice quantitative results.

3.13.10 Example: Excited States

As a final example, we do a state-average calculation on H_2CO in order to illustrate excited state treatments. We expect from the ground state (basically closed-shell) a n $\rightarrow \pi^*$ and a $\pi \rightarrow \pi^*$ excited state which we want to describe. For the n$\rightarrow \pi^*$ we also want to calculate the triplet since it is well known experimentally. First we take DFT orbitals as starting guess.

```
! BP86 Def2-SVP TightSCF

*int 0 1
 C  0 0 0 0.00   0.0    0.00
 O  1 0 0 1.20   0.0    0.00
 H  1 2 0 1.10 120.0    0.00
 H  1 2 3 1.10 120.0  180.00
end
```

In this example the DFT calculation produces the desired active space (n,π andπ^* orbitals) without further modification (e.g. swapping orbitals). In general it is advised to verify the final converged orbitals.

```
! Def2-SVP TightSCF MOREAD

%moinp "orbs.gbw"

%casscf
 nel         4
 norb        3
 mult        1,3
 nroots      3,1
end

*int 0 1
C  0 0 0 0.00   0.0    0.00
O  1 0 0 1.20   0.0    0.00
H  1 2 0 1.10 120.0    0.00
H  1 2 3 1.10 120.0  180.00
end
```

We get:

```
-----------------------------
SA-CASSCF TRANSITION ENERGIES
-----------------------------

LOWEST ROOT (ROOT 0 ,MULT 1) =   -113.805194041 Eh -3096.797 eV

STATE   ROOT MULT  DE/a.u.    DE/eV    DE/cm**-1
1:    0    3   0.129029     3.511  28318.5
2:    1    1   0.141507     3.851  31057.3
3:    2    1   0.453905    12.351  99620.7
```

The triplet n $\rightarrow$ π^* states is spot on with the experiment excitation energy of 3.5 eV.[451] Similarly, the singlet n $\rightarrow$ π^* excited state is well reproduced compared to 3.79 eV and 4.07 eV reported in the literature.[451, 452] Only the singlet π $\rightarrow$ π^* excited state stands out compared to the theoretical estimate of 9.84 eV computed with MR-AQCC.[453]. The good results are very fortuitous given the small basis set, the minimal active space and the complete neglect of dynamical correlation.

The state-average procedure might not do justice to the different nature of the states (n $\rightarrow$ π^* versus π $\rightarrow$ π^*). The agreement should be better with the orbitals optimized for each state. In ORCA, state-specific optimization are realized adjusting the weights i.e. for the second singlet excited root:

```
Second-Singlet:
%casscf
  # define CAS(5,4)
  nel         4
  norb        3
  # define singlet with and compute the 3 lowest
  # states
  mult        1
  nroots      3
  weights[0] = 0,0,1 # weights for the roots
end
```

Note, that state-specific orbital optimization are challenging to converge and often prone to root-flipping.[454]

To analyze electronic transitions, natural transition orbitals (NTO) are available for state-averaged CASSCF (and also CASCI) calculations. NTOs are switched on for every ground- to excited-state transition by just adding `DoNTO true` to the `%casscf ... end` input block, i.e.

```
%casscf
  # define CAS(4,3)
  nel    4
```

(continues on next page)

(continued from previous page)

```
  norb   3

  # state-average
  mult   1,3
  nroots 3,1

  DoNTO  true
end
```

For each excitation, the most dominant natural occupation numbers (singular values >1.e-4) are printed for each transition. A set of donor orbitals and a set of acceptor orbitals, each of dimension Nbf x (Nocc + Nact), are created and stored in files with unique names. We obtain for the previous formamide example the following CASSCF NTO output

```
=========================================
CASSCF Natural Transition Orbitals
=========================================

-------------------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE     1 1A
-------------------------------------------------

STATE    1 1A  :  E=   0.141508 au      3.851 eV    31057.3 cm**-1

Threshold for printing occupation numbers 1.0000e-04

     0  : n=  1.30882812
     1  : n=  0.02641080

=> Natural Transition Orbitals (donor   ) were saved in Test-CASSCF.H2CO-1.casscf.
↪1-1A_nto-donor.gbw
=> Natural Transition Orbitals (acceptor) were saved in Test-CASSCF.H2CO-1.casscf.
↪1-1A_nto-acceptor.gbw

-------------------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE     2 1A
-------------------------------------------------

STATE    2 1A  :  E=   0.453905 au     12.351 eV    99620.7 cm**-1

Threshold for printing occupation numbers 1.0000e-04

     0  : n=  1.30519478
     1  : n=  0.24869813
     2  : n=  0.00471742

=> Natural Transition Orbitals (donor   ) were saved in Test-CASSCF.H2CO-1.casscf.
↪2-1A_nto-donor.gbw
=> Natural Transition Orbitals (acceptor) were saved in Test-CASSCF.H2CO-1.casscf.
↪2-1A_nto-acceptor.gbw
```

For each transition, plots of the NTO pairs can be generated with the `orca_plot` program (see Sec. *Orbital and Density Plots* for details), , e.g., acceptor orbitals of the 2 1A1 state in interactive mode:

```
orca_plot Test-CASSCF.H2CO-1.casscf.2-1A_nto-acceptor.gbw -i
```

Fig. 3.21: Most dominant natural transition orbital (NTO) pair for the 2 1A1 (S2) transition in formaldehyde.

Alternatively, NTOs can also be computed directly in `orca_plot` from the CASSCF transition density matrices. Those need to be stored and kept in the density container by invoking

```
! KeepTransDensity
```

3.13.11 Example: Large Active Spaces using ICE-CI

The state-averaged CASSCF procedure can be used for the calculation of spin-state energetics of molecules showing a multi-reference character. The main obstacle in getting qualitatively accurate spin-state energetics for molecules with many transition metal centers is the proper treatment of the static-correlation effects between the large number of open-shell electrons. In this section, we describe how one can effectively perform CASSCF calculations on such systems containing a large number of high-spin open-shell transition metal atoms.

As an example, consider the Iron-Sulfur dimer $[\mathrm{Fe(III)}_2\mathrm{SR}_2]^{2-}$ molecule. In this system, the Fe(III) centers can be seen as being made up mostly of S=5/2 local spin states (lower spin states such as 3/2 and 1/2 will have small contributions due to Hunds' rule.) The main hurdle while using the CASSCF protocol on such systems (with increasing number of metal atoms) is the exponential growth of the Hilbert space although the physics can be effectively seen as occurring in a very small set of configuration state functions (CSFs). Therefore, in order to obtain qualitatively correct spin-state energetics, one need not perform a Full-CI on such molecules but rather a CIPSI like procedure using the ICE-CI solver should give chemically accurate results. In the case of the Fe(III) dimer, one can imagine that the ground singlet state is composed almost entirely of the CSF where the two Fe(III) centers are coupled antiferromagnetically. Such a CSF is represented as follows:

$$\left|\Psi_0^{S=0}\right\rangle = [1, 1, 1, 1, 1, -1, -1, -1, -1, -1]$$

In order to make sense of this CSF representation, one needs to clarify a few points which are as follows:

- First, in the above basis the 10 orbitals are localized to 5 on each Fe center (following a high-spin UHF/UKS calculation.)
- Second, the orbitals are ordered (as automatically done in ORCA_LOC) such that the first five orbitals lie on one Fe(III) center and the last five orbitals on the second Fe(III) center.

Using this ordering, one can read the CSF shown above in the following way: The first five *1* represent the five electrons on the first Fe(III) coupled in a parallel fashion to give a S=5/2 spin. The next five *-1* represent two points:

- First, the five consecutive *-1* signify the presence of five ferromagnetically coupled electrons on the second Fe(III) center resulting in a local S=5/2 spin state.
- Second, the second set of spins are coupled to the first *1* via anti-parallel coupling as signified by the sign of the last five *-1* entries.

Therefore, we can see that using the CSF representation, one can obtain an extremely compact representation of the wave function for molecules consisting of open-shell transition metal atoms. This protocol of using localized orbitals in a specified order to form compact CSF representations for transition metal systems can be systematically extended for large molecules.

We will use the example of the Iron-Sulfur dimer $[\mathrm{Fe(III)}_2\mathrm{SR}_2]^{2-}$ to demonstrate how to prepare a reference CSF and perform spin-state energetics using the state-averaged CASSCF protocol. In such systems, often one can obtain

an estimate of the energy gap between the singlet-state and the high-spin states from experiment. Ab initio values for this gap be obtained using the state-averaged CASSCF protocol using the input shown below.

```
! def2-SVP MOREAD

%moinp "locorbs.gbw"

%casscf
 nel     10
 norb    10
 mult    11,1
 nroots  1,1
 refs                              # reference for multiplicity 11
  { 1 1 1 1 1  1  1  1  1  1}
 end
 refs                              # reference for multiplicity 1
  { 1 1 1 1 1 -1 -1 -1 -1 -1}
 end
  cistep ice
 ci
  icetype 1
 end
 actorbs unchanged
end

* xyz -2 11
Fe          0.000000000      0.000000000     -1.343567812
Fe          0.000000000      0.000000000      1.343567812
S           1.071733501      1.373366082      0.000000000
S           1.346714284     -1.345901486     -2.651621449
S          -1.346714284      1.345901486     -2.651621449
S          -1.071733501     -1.373366082      0.000000000
S          -1.346714284      1.345901486      2.651621449
S           1.346714284     -1.345901486      2.651621449
C          -2.485663304      0.362543393     -3.600795276
H          -3.319937516      0.596731755     -3.505882795
H          -2.347446507      0.388292903     -4.463380590
H          -2.472404709     -0.485711203     -3.404167343
C           2.485663304     -0.362543393     -3.600795276
H           3.319937516     -0.596731755     -3.505882795
H           2.347446507     -0.388292903     -4.463380590
H           2.472404709      0.485711203     -3.404167343
C           2.485663304     -0.362543393      3.600795276
H           2.347446507     -0.388292903      4.463380590
H           3.319937516     -0.596731755      3.505882795
H           2.472404709      0.485711203      3.404167343
C          -2.485663304      0.362543393      3.600795276
H          -3.319937516      0.596731755      3.505882795
H          -2.472404709     -0.485711203      3.404167343
H          -2.347446507      0.388292903      4.463380590
*
```

The main keyword that needs to be used here (unlike in other CAS-SCF calculations) is the *actorbs* keyword. Since we are using a local basis with a specific ordering of the orbitals, in order to represent our wave function it is imperative to preserve the local nature of the orbitals as well as the orbital ordering. Therefore, we do not calculate natural orbitals at the end of the CASSCF calculation (as is traditionally done) instead we impose the orbitals to be as similar to the input orbitals as possible. This is automatically enabled for intermediate CASSCF macro iterations. The resulting CASSCF calculation provides a chemically intuitive and simple wave function and transition energy as shown below:

```
---------------------------------------------
CAS-SCF STATES FOR BLOCK  1 MULT=11 NROOTS= 1
---------------------------------------------
```

(continues on next page)

(continued from previous page)

```
STATE   0 MULT=11: E=  -5066.8462457411 Eh W=  0.5000 DE= 0.000 eV      0.0 cm**-1
      1.00000 ( 1.000000000) CSF = 1+1+1+1+1+1+1+1+1+1+



---------------------------------------------
CAS-SCF STATES FOR BLOCK  2 MULT= 1 NROOTS= 1
---------------------------------------------

STATE   0 MULT= 1: E=  -5066.8548894831 Eh W=  0.5000 DE= 0.000 eV      0.0 cm**-1
      0.98159 (-0.990753235) CSF = 1+1+1+1+1+1-1-1-1-1-



-----------------------------
SA-CASSCF TRANSITION ENERGIES
-----------------------------

LOWEST ROOT (ROOT 0 ,MULT 1) =  -5066.854889483 Eh -137876.131 eV

STATE   ROOT MULT  DE/a.u.    DE/eV    DE/cm**-1
   1:    0   11   0.008644     0.235   1897.1
```

As we can see from the output above, 98% of the wave function for the singlet-state is given by a single CSF which we gave as a reference CSF. This CSF has a very simple chemical interpretation representing the anti-parallel coupling between the two high-spin Fe(III) centers. Since Iron-Sulfur molecules show a strong anti-ferromagnetic coupling, we expect the singlet state to be lower in energy than the high-spin (S=5) state. The CASSCF transition energies show essentially this fact. The transition energy is about 2000cm^{-1} which is what one expects from a CASSCF calculation on such sulfide bridged transition-metal molecules.

3.13.12 Example: Geometry Optimization

In this section, we want to optimize the geometry of formaldehyde with CASSCF. The present implementation of analytic gradients are only implemented in the case that the solution is fully variational (orbitals are optimized for a single state).

To define the active space, lets first look at the natural orbitals from a simple correlation calculation like MP2 or a calculation with the MRCI module. Both are usually a good choice and easily generated. For example:

```
#
# First job provides reasonable natural orbitals
#
! RI-MP2 SVP def2-SVP/C

%mp2 natorbs true
     density unrelaxed # or relaxed (more expensive)
     end

* int 0 1
    C  0 0 0 0.00   0.0    0.00
    O  1 0 0 1.20   0.0    0.00
    H  1 2 0 1.10 120.0    0.00
    H  1 2 3 1.10 120.0  180.00
*
```

Now examine the occupation numbers of the natural orbitals (you will find that in the output of the MP2 part of the calculation):

```
Natural Orbital Occupation Numbers:
N[  0] =   2.00000000
N[  1] =   2.00000000
N[  2] =   1.98676733
N[  3] =   1.97726840
N[  4] =   1.97500109
N[  5] =   1.96759239
N[  6] =   1.96423113
N[  7] =   1.93719340
N[  8] =   0.05427454
N[  9] =   0.02555886
N[ 10] =   0.02530580
N[ 11] =   0.01358500
N[ 12] =   0.01096092
N[ 13] =   0.01028129
N[ 14] =   0.00702048
N[ 15] =   0.00627820
```

A rule of thumb is that orbitals with occupation numbers between 1.98 and 0.02 should be in the active space. Thus, in the present case we speculate that a 10 electrons in 8 orbitals active space would be appropriate for the CASSCF of the ground state. Let's try:

```
#
# Run a CASSCF calculation for the ground state of H2CO
#
! SVP def2-SVP/C SmallPrint
! moread
# reading mp2 natural orbitals
%moinp "Test-CASSCF-MP2-H2CO.mp2nat"

%casscf
        # define CAS(10,8)
        nel        10
        norb        8
        # singlet ground state
               mult        1
        nroots      1
end

* int 0 1
    C  0 0 0 0.00   0.0    0.00
    O  1 0 0 1.20   0.0    0.00
    H  1 2 0 1.10 120.0    0.00
    H  1 2 3 1.10 120.0  180.00
*
```

If we run that calculation, it converges and produces the following:

```
  MACRO-ITERATION  10:
  --- Inactive Energy E0 = -82.97337099 Eh
  E(CAS)=  -113.889438276 Eh DE=    -0.000000807
  --- Energy gap subspaces: Ext-Act = -0.431   Act-Int = -0.240
  N(occ)=  1.99763 1.99696 1.98360 1.97923 1.94253 0.05958 0.02153 0.01894
  ||g|| =      0.000361782 Max(G)=     0.000189613 Rot=9,2
  ---- THE CAS-SCF GRADIENT HAS CONVERGED ----
            --- FINALIZING ORBITALS ---
  ---- DOING ONE FINAL ITERATION FOR PRINTING ----
  --- Forming Natural Orbitals
  --- Canonicalize Internal Space
  --- Canonicalize External Space
```

From which we see that we had *two orbitals too many in the active space* with occupation numbers very close to two.

The presence of barely correlated orbitals (occupation close to 0.0 or 2.0) can cause convergence problems. Their inclusion in the active space does not significantly change the energy and it might better to omit these orbitals from the start.

In the present case, we re-run the CASSCF with 6 active electrons in six orbitals. The result is:

```
...
MACRO-ITERATION   8:
--- Inactive Energy E0 = -100.52457962 Eh
E(CAS)=  -113.885584276 Eh DE=    -0.000001152
--- Energy gap subspaces: Ext-Act = -0.438   Act-Int = -0.283
N(occ)=  1.98134 1.97931 1.94184 0.05868 0.02101 0.01781
||g|| =      0.000752012 Max(G)=    -0.000357510 Rot=13,4
---- THE CAS-SCF GRADIENT HAS CONVERGED ----
--- FINALIZING ORBITALS ---
---- DOING ONE FINAL ITERATION FOR PRINTING ----
--- Forming Natural Orbitals
--- Canonicalize Internal Space
--- Canonicalize External Space

MACRO-ITERATION   9:
--- Inactive Energy E0 = -100.52457962 Eh
--- All densities will be recomputed
E(CAS)=  -113.885584276 Eh DE=    -0.000000000
--- Energy gap subspaces: Ext-Act = -0.858   Act-Int = -0.283
N(occ)=  1.98172 1.97932 1.94207 0.05845 0.02100 0.01743
||g|| =      0.000752012 Max(G)=    -0.000327367 Rot=12,4
--------------
CASSCF RESULTS
--------------

Final CASSCF energy       : -113.885584276 Eh   -3098.9843 eV
```

The calculation converges very quickly and the occupation numbers show you that all of these orbitals are actually needed in the active space. The omission of the two orbitals from the active space came at an increase of the energy by ∼4 mEh which seems to be tolerable. Let's look what we have in the active space in figure Fig. 3.22.

(a) MO5 (b) MO6 (c) MO7

(d) MO10 (e) MO9 (f) MO8

Fig. 3.22: Orbitals of the active space for the CASSCF(6,6) calculation of H_2CO.

Thus, we can see that we got a fairly nice result: our calculation has correlated the in-plane oxygen lone pair, the C-O σ and the C-O π bond. For each strongly occupied bonding orbital, there is an accompanying weakly occupied antibonding orbital in the active space that is characterized by one more node. In particular, the correlating lone pair and the C-O σ^* orbital would have been hard to find with any other procedure than the one chosen based on natural orbitals. We have now done it blindly and looked at the orbitals only after the CASSCF — a better approach is normally to look at the starting orbitals *before* you enter a potentially expensive CASSCF calculation. If you have bonding/antibonding pairs in the active space plus perhaps the singly-occupied MOs of the system you probably have chosen a reasonable active space.

Now that we establish the good active space, let us optimize the geometry of the molecule using a reasonable basis set:

```
! def2-TZVP def2-TZVP/C SmallPrint Opt
! moread
%moinp "Test-CASSCF-MP2-H2CO.mp2nat"

%casscf nel         6
        norb        6
        end

* int 0 1
    C  0 0 0 0.00   0.0    0.00
    O  1 0 0 1.20   0.0    0.00
    H  1 2 0 1.10 120.0    0.00
    H  1 2 3 1.10 120.0  180.00
*
```

and get:

```
-----------------------------------------------------------------------------
    Redundant Internal Coordinates

    --- Optimized Parameters ---
    (Angstroem and degrees)

    Definition                    OldVal   dE/dq     Step     FinalVal
    ---------------------------------------------------------------------------
    1. B(O   1,C   0)               1.2101  0.000259 -0.0002    1.2100
    2. B(H   2,C   0)               1.0942 -0.000029  0.0001    1.0943
    3. B(H   3,C   0)               1.0942 -0.000029  0.0001    1.0943
    4. A(O   1,C   0,H   3)         122.07  0.000023   -0.00    122.07
    5. A(H   2,C   0,H   3)         115.85 -0.000046    0.01    115.86
    6. A(O   1,C   0,H   2)         122.07  0.000023   -0.00    122.07
    7. I(O   1,H   3,H   2,C   0)     -0.00 -0.000000    0.00     -0.00
    ---------------------------------------------------------------------------
```

Let us compare to MP2 geometries (this job was actually run first):

```
! RI-MP2 def2-TZVP def2-TZVP/C Opt

%mp2 natorbs true
     end

* int 0 1
    C  0 0 0 0.00   0.0    0.00
    O  1 0 0 1.20   0.0    0.00
    H  1 2 0 1.10 120.0    0.00
    H  1 2 3 1.10 120.0  180.00
*
```

```
-----------------------------------------------------------------------
                        Redundant Internal Coordinates

                         --- Optimized Parameters ---
                           (Angstroem and degrees)

        Definition                    OldVal   dE/dq     Step     FinalVal
    ---------------------------------------------------------------------------
     1. B(O   1,C   0)               1.2127  0.000374 -0.0002    1.2125
     2. B(H   2,C   0)               1.0991 -0.000031  0.0001    1.0992
     3. B(H   3,C   0)               1.0991 -0.000031  0.0001    1.0992
     4. A(O   1,C   0,H   3)         121.77  0.000023   -0.00    121.77
     5. A(H   2,C   0,H   3)         116.45 -0.000046    0.01    116.46
     6. A(O   1,C   0,H   2)         121.77  0.000023   -0.00    121.77
     7. I(O   1,H   3,H   2,C   0)     -0.00 -0.000000    0.00     -0.00
    ---------------------------------------------------------------------------
```

The results are actually extremely similar (better than 1 pm agreement). Compare to RHF:

```
-----------------------------------------------------------------------
                        Redundant Internal Coordinates

                         --- Optimized Parameters ---
                           (Angstroem and degrees)

        Definition                    OldVal   dE/dq     Step     FinalVal
    ---------------------------------------------------------------------------
     1. B(O   1,C   0)               1.1784 -0.000164  0.0001    1.1785
     2. B(H   2,C   0)               1.0921  0.000010 -0.0000    1.0921
     3. B(H   3,C   0)               1.0921  0.000010 -0.0000    1.0921
     4. A(O   1,C   0,H   3)         121.93 -0.000003   -0.00    121.93
```

(continues on next page)

(continued from previous page)

```
     5. A(H   2,C   0,H   3)          116.13  0.000005    0.00    116.13
     6. A(O   1,C   0,H   2)          121.93 -0.000003   -0.00    121.93
     7. I(O   1,H   3,H   2,C   0)      0.00  0.000000   -0.00     -0.00
    ----------------------------------------------------------------------
```

Thus, one can observe that the correlation brought in by CASSCF or MP2 has an important effect on the C=O distance (∼4 pm), while the rest of the geometry is not much affected.

3.13.13 Example: Natural Orbitals as Input for Coupled-Cluster Calculations

Consider the possibility that you are not sure about the orbital occupancy of your system. Hence you carry out some CASSCF calculation for various states of the system in an effort to decide on the ground state. You can of course follow the CASSCF by MR-MP2 or MR-ACPF or SORCI calculations to get a true multireference result for the state ordering. Yet, in some cases you may also want to obtain a coupled-cluster estimate for the state energy difference. Converging coupled-cluster calculation on alternative states in a controlled manner is anything but trivial. Here a feature of ORCA might be helpful. The best single configuration that resembles a given CASSCF state is built from the natural orbitals of this state. These orbitals are also ordered in the right way to be input into the MDCI program. The convergence to excited states is, of course, not without pitfalls and limitations as will become evident in the two examples below.

As an example, consider some ionized states of the water cation:

First, we generate the natural orbitals for each state with the help of the MRCI module. To this end we run a state average CASSCF for the lowest three doublet states and pass that information on to the MRCI module that does a CASCI calculation and produces the natural orbitals:

```
! ano-pVDZ TightSCF

%casscf
 nel     7
 norb    6
 nroots  3
 mult    2
end

%mrci
 tsel       0
 tpre       0
 donatorbs  2
 densities  5,1
 newblock 2 *
  nroots 3
  excitations none
  refs
   cas(7,6)
  end
 end
end

* int 1 2
 O     0   0   0   0.000000     0.000     0.000
 H     1   0   0   1.012277     0.000     0.000
 H     1   2   0   1.012177   109.288     0.000
end
```

This produces the files `Basename.bm_sn.nat` where “m” is the number of the block (m = 0 correspond to the doublet in this case) and “n” stands of the relevant state (n = 0,1,2).

These natural orbitals are then fed into unrestricted QCISD(T) calculations:

```
! ano-pVDZ TightSCF QCISD(T) MOREAD NoIter

%moinp "H2O+.b0_s0.nat"

* int 1 2
 O     0   0   0   0.000000     0.000     0.000
 H     1   0   0   1.012277     0.000     0.000
 H     1   2   0   1.012177   109.288     0.000
*
```

As a reference we also perform a SORCI on the same system

```
! ano-pVDZ TightSCF SORCI

%casscf
 nel     7
 norb    6
 nroots  3
 mult    2
end

* int 1 2
 O     0   0   0   0.000000     0.000     0.000
 H     1   0   0   1.012277     0.000     0.000
 H     1   2   0   1.012177   109.288     0.000
*
```

we obtain the transition energies:

```
      SORCI   QCISD(T) (in cm-1)
  D0      0     0.0
  D1  16269   16293
  D2  50403   50509
```

Thus, in this example the agreement between single- and multireference methods is good and the unrestricted QCISD(T) method is able to describe these excited doublet states. The natural orbitals have been a reliable way to guide the CC equations into the desired solutions. This will work in many cases.

3.13.14 CASSCF Densities

In general, densities are stored in the so-called density container (`.densities`file). Using the *`orca_plot` program* we can print a list of the available densities:

```
orca_plot jobname.densities
```

By default, the state-averaged density ('.scfp') and spin-density ('.scfr') are stored. With the keyword `!KeepDens`, ORCA stores all of the computed densities. Below is an example snippet from a *QD-NEVPT2 calculation* with spin-orbit coupling.

```
---------------------
List of density names
---------------------

Index:                                                  Name of Density
-----------------------------------------------------------------------
    0:                                                         cas.scfp
    1:                                                         cas.scfr
    2:                                          Tdens-CAS.mult.3.root.0.p
    3:                                          Tdens-CAS.mult.3.root.1.p
    4:                                          Tdens-CAS.mult.3.root.2.p
```

(continues on next page)

(continued from previous page)

```
  5:                                        Tdens-CAS.mult.1.root.0.p
  6:                                        Tdens-CAS.mult.1.root.1.p
  7:                                        Tdens-CAS.mult.1.root.2.p
  8:                                        Tdens-CAS.mult.3.root.0.r
  9:                                        Tdens-CAS.mult.3.root.1.r
 10:                                        Tdens-CAS.mult.3.root.2.r
 11:                                        Tdens-CAS.mult.1.root.0.r
 12:                                        Tdens-CAS.mult.1.root.1.r
 13:                                        Tdens-CAS.mult.1.root.2.r
 14:                              Tdens-CAS-DPDICPT2.mult.3.root.0.r
 15:                              Tdens-CAS-DPDICPT2.mult.3.root.1.r
 16:                              Tdens-CAS-DPDICPT2.mult.3.root.2.r
 17:                              Tdens-CAS-DPDICPT2.mult.1.root.0.r
 18:                              Tdens-CAS-DPDICPT2.mult.1.root.1.r
 19:                              Tdens-CAS-DPDICPT2.mult.1.root.2.r
 20:                                        Tdens-CASQDSOCMO.root.0.p
 21:                                          Tdens-CASQDSOC.root.0.p
 22:                                     Tdens-CASPTQDSOCMO.root.0.p
 23:                                        Tdens-CASPTQDSOC.root.0.p
 24:                              Tdens-CAS-DPDICPT2QDSOCMO.root.0.p
 25:                                Tdens-CAS-DPDICPT2QDSOC.root.0.p
```

Here, the densities 2-12 are CASSCF state-specific densities denoted with "Tdens-CAS" and a string flagging its multiplicity (potentially also irreducible representation) and root count. Similarly, densities 13-19 arise from the QD-NEVPT2 procedure and carry the prefix "Tdens-CAS-DPDICPT2". The latter is shared with other diagonalize-perturb-diagonalize type theories (DPD). The remaining densities containing the "SOC" substring, belong to densities computed in the framework of *quasi-degenerate peturbation theory* and differ in the handling of the reference wavefunction and the choice of the diagonal energies. The prefix "Tdens-CASQDSOC" refers to the canonical CASSCF results. Densities with the NEVPT2 diagonal energies are maked as "TDens-CASPTQDSOC". And finally the density using the QD-NEVPT2 revised reference wavefunction contain the string "Tdens-CAS-DPDICPT2QDSOC".

Warning

Densities with the "MO" substring in the prefix should be ignored as they are intermediates.

3.13.15 CASSCF Properties

The CASSCF program is able to calculate UV transition, CD spectra, SOC, SSC, Zeeman splittings, EPR g-matrices and A-matrices (the latter implemented in the same way as in the DCD-CAS(2) method[432]), magnetization, magnetic susceptibility and MCD spectra. Note that the results for the Fermi contact contribution to A will not be reliable if the spin density is dominated by spin polarization, which is a dynamic correlation effect. The properties are exercised in more detail in the CASSCF tutorial.

In ORCA, the UV/CD spectra are computed using via the *one photon spectrocopy tool*. A few selected properties, such as the Zeeman splitting, are available through linear response described in the section *CASSCF Linear Response*. The majority of magnetic properties are computed in the framework of *quasi-degenrate perturbation theory*, which is also used in the `orca_mrci` program. Input and keywords mimic the ones in the MRCI module described in section Section 3.19.14. As an example, the input file to calculate g-values and HFC constants A of CO^+ is listed below:

```
!TZVPP  Bohrs TightSCF #TightSCF for more accurate integrals
%casscf nel  9
        norb 8
        nroots 9
        mult   2
        switchstep NR
```

(continues on next page)

(continued from previous page)

```
        etol 1e-7 #reset energy convergence
        rel
            dosoc true   #spin-orbit coupling (and ZFS)
            gtensor true
            amatrix true
            amatrixnuc 0,1
        end
        end
* xyz 1 2
 C 0 0  0.0
 O 0 0  2.3504
*
```

In addition to pseudo-spin 1/2 A-tensors for individual Kramers doublets, the CASSCF module also features the calculation of "intrinsic" HFC A-tensors for the whole lowest nonrelativistic spin multiplet in the effective Hamiltonian approach.[455]

In contrast to the MRCI module, the CASSCF module also supports the calculation of susceptibility tensors at non-zero magnetic fields. The corresponding keywords are

```
...
%casscf
  ...
  rel
    dosoc true
    dosusceptibility true
    susctensor_nfields 2                    # number of user-defined magnetic␣
↪fields
    susctensor_magfields[0] = 35000,0,0     # 1st user-defined magnetic field
    susctensor_magfields[1] = 70000,0,0     # 2nd user-defined magnetic field
  end
end
```

This example input calculates the susceptibility tensor at the two (vector-valued!) magnetic fields (35000,0,0) and (70000,0,0) (in Gauss). Note that for practical reasons it is necessary to specify the number of user-defined magnetic fields using the keyword `susctensor_nfields`.

Until ORCA 4.0 it was possible to access spin-spin couplings only via running CAS-CI type calculations in MRCI. Converged CASSCF orbitals can be read setting the following flags

```
!MOREAD NOITER ALLOWRHF TZVPP  TightSCF Bohrs
%moinp "convergedCASSCF.gbw"

%mrci
  ...
  TPre 0.0
  citype mrci

  newblock 2 *
    excitations none
    refs CAS(9,8) end
  end

  soc
    DoSSC true # spin-spin coupling
    DoSOC true # spin-orbit coupling
    ...
  end
end
* xyz 1 2
 C 0 0  0.0
```

(continues on next page)

(continued from previous page)

```
 O 0 0  2.3504
*
```

Starting with ORCA 4.1, spin-spin couplings are also directly accessible in the CASSCF module via the keyword `DoSSC true` in the `rel` subblock. Note that the calculation of SSC requires the definition of an auxiliary basis set (`AuxC` auxiliary basis set slot), since it is only implemented in conjunction with RI integrals. A common way to introduce dynamical correlation for the property computation, is to replace the energies entering the quasi-degenerate perturbation theory. If the NEVPT2 energy correction is computed in CASSCF, there will be additional printings where CASSCF energies are replaced by the more accurate NEVPT2 values. Alternatively, these diagonal energies can be taken from the input file similarly how it is described for the MRCI module. A more detailed documentation is presented in the MRCI property section.

Note

- The program does NOT print the SOC matrix by default! To obtain SOCMEs at the CASSCF/NEVPT2/… levels, please set the `PrintLevel` in the `rel` block to at least 2.

3.13.16 1- and 2-shell Abinitio Ligand Field Theory (AILFT)

Starting from ORCA 5.0, ORCA features a 1- and 2-shell AILFT module. AILFT was originally developed for 1-shell d- and f- LFT problems [433, 434]. In ORCA 5.0 an extenion to 2-shell AILFT was devloped by increasing the parameter space of the LFT hamiltonian. This provides access to all common 1- and 2-shell AILFT problems namely:

1. Valence LFT problems, involving the d-, f-, sp-, ds- and df-shells
2. Core LFT problems involving the sd-, pd-, sf- and pf-shells become readily accesible.

(Note: Since ORCA 6.1 it is also possible to project the ab initio hamiltonian of an extended active space calculation onto the 1-shell AILFT problem. This way one recover improved 1-shell parameters without introducing new parameters. This is covered in detail in the subsequent section.)

Requesting an CAS-AILFT calcultion withing the CASSCF module is provided in two ways:

1. Through the ActOrbs xOrbs keywords (e.g. xOrbs: dOrbs, fOrbs spOrbs, psOrbs, sdOrbs, dsOrbs, sfOrbs, fsOrbs, pdOrbs, dpOrbs, pfOrbs, fpOrbs, dfOrbs, fdOrbs)
2. Through the LFTCase keyword where particular LFT problems can be requested according to the above 1- and 2-shell combinations (e.g. LFTCase 3d, LFTCase 4f, LFTCase 1s3d, LFTCase 2p3d …)

Note: that the LFTCase keyword overwrites the ActOrbs keyword and as it will be discussed below provides a particular utility that simplifies the 2-shell AILFT input.

A simple input for the Ni^{2+} d^8 ion is provided below:

```
!NEVPT2 def2-SVP def2-SVP/C
%casscf
 nel 8
 norb 5
 ActOrbs dOrbs
 mult 3,1
 nroots 10,15
 rel
  dosoc true
 end
end

*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

The programm after the CASSCF convergence will undergo few important steps and sanity checks which involve

1. an Orbital purification step
2. a Phase correction of the 1 and 2-electron integrals

It is then important from the user's perspective to monitor that these steps have been succesfully performed. The relevant parts of the output are provided below:

```
---- THE CAS-SCF GRADIENT HAS CONVERGED ----
--- FINALIZING ORBITALS ---
---- DOING ONE FINAL ITERATION FOR PRINTING ----
--- d-orbitals (depends on the molecular axis frame)
L-Center:  0 Ni [0.000, 0.000, 0.000]
--- The active space contains 5 d Orbitals : OK
Setting 9 active MO to AO dz2     (11)
Setting 10 active MO to AO dxz     (12)
Setting 11 active MO to AO dyz     (13)
Setting 12 active MO to AO dx2y2   (14)
Setting 13 active MO to AO dxy     (15)
--- Canonicalize Internal Space
--- Canonicalize External Space

...


============================
AB INITIO LIGAND FIELD THEORY
d8 configuration
2 CI blocks
MOs 9 to 13
============================

Metal/Atom center is atom 0
orbital phases =   1.0   1.0   1.0   1.0   1.0
Metal/Atom d-orbital parts of active orbitals
Shell 7
9        10        11        12        13
dz2     :  0.848522 -0.000000  0.000000 -0.000000 -0.000000
dxz     : -0.000000  0.848522  0.000000 -0.000000  0.000000
dyz     :  0.000000  0.000000  0.848522 -0.000000 -0.000000
dx2y2   : -0.000000 -0.000000 -0.000000  0.848522  0.000000
dxy     : -0.000000  0.000000 -0.000000  0.000000  0.848522
Shell 8
9        10        11        12        13
dz2     :  0.300072  0.000000 -0.000000  0.000000  0.000000
dxz     :  0.000000  0.300072 -0.000000  0.000000 -0.000000
dyz     : -0.000000 -0.000000  0.300072  0.000000  0.000000
dx2y2   :  0.000000  0.000000  0.000000  0.300072 -0.000000
dxy     :  0.000000 -0.000000  0.000000 -0.000000  0.300072

Adjusting phases of one-electron integrals               ... done
Adjusting phases of two-electron integrals               ... done
```

In a subsequent step the program will

1. compute the AI Hamiltonian
2. construct the parameterized LFT Hamiltonian
3. and perform the fit

The relevant output can be seen below:

```
Calculating ab initio Hamiltonian matrices                ...
------------------------------------------------------------
NRoots (NEVPT2) for this block = 10
NEVPT2 correction for this block is calculated
Full NEVPT2 Hamiltonian constructed
Full NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
NRoots (NEVPT2) for this block = 15
NEVPT2 correction for this block is calculated
Full NEVPT2 Hamiltonian constructed
Full NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
Calculating fit matrices                                  ... done
Fitting                                                   ... done
```

In following the fitted 1-electron energies and SCP parameters also Racah parameters for 1-shells will be printed at the CASSCF and NEVPT2 levels of theory

```
-----------------------------
AILFT MATRIX ELEMENTS (CASSCF)
--------------------------------

Ligand field one-electron matrix VLFT (a.u.) :
Orbital          dz2        dxz        dyz        dx2-y2     dxy
dz2         -8.111733  -0.000000  -0.000000   0.000000  -0.000000
dxz         -0.000000  -8.111733  -0.000000  -0.000000   0.000000
dyz         -0.000000  -0.000000  -8.111733  -0.000000   0.000000
dx2-y2       0.000000  -0.000000  -0.000000  -8.111733  -0.000000
dxy         -0.000000   0.000000   0.000000  -0.000000  -8.111733

--------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
--------------------------------------------------
F0dd(from 2el Ints)     =  0.980960738 a.u. =    26.693 eV =  215296.0 cm**-1␣
↪(fixed)
F2dd                    =  0.451725025 a.u. =    12.292 eV =   99142.2 cm**-1
F4dd                    =  0.280604669 a.u. =     7.636 eV =   61585.6 cm**-1
-------------------
Racah Parameters :
-------------------
A(F0dd from 2el Ints)   =  0.949782441 a.u. =    25.845 eV =  208453.2 cm**-1
B                       =  0.006037419 a.u. =     0.164 eV =    1325.1 cm**-1
C                       =  0.022270212 a.u. =     0.606 eV =    4887.7 cm**-1
C/B                     =  3.689
-------------------------------------------------------------------------------

----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV)  Energy(cm-1)      dz2        dxz        dyz        dx2-y2 ␣
↪   dxy
1          0.000        0.0        -0.999978  -0.000164  -0.001934   0.005783  -0.
↪002568
2          0.000        0.0        -0.005768  -0.000269  -0.000262  -0.999967  -0.
↪005788
3          0.000        0.0         0.002600   0.000424   0.001046   0.005773  -0.
↪999979
4          0.000        0.0         0.000241  -0.999246  -0.038831   0.000280  -0.
↪000462
5          0.000        0.0         0.001930   0.038832  -0.999243   0.000246  -0.
↪001022
```

(continues on next page)

(continued from previous page)

```
Ligand field orbitals were stored in ni.3d.casscf.lft.gbw

...

-----------------------------
AILFT MATRIX ELEMENTS (NEVPT2)
-------------------------------

Ligand field one-electron matrix VLFT (a.u.) :
Orbital          dz2        dxz        dyz        dx2-y2     dxy
dz2         -8.118685   0.000000   0.000000   0.000005  -0.000000
dxz          0.000000  -8.118666  -0.000000  -0.000000   0.000000
dyz          0.000000  -0.000000  -8.118674  -0.000000   0.000000
dx2-y2       0.000005  -0.000000  -0.000000  -8.118676   0.000000
dxy         -0.000000   0.000000   0.000000   0.000000  -8.118667

--------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
--------------------------------------------------
F2dd                    =  0.415943380 a.u. =    11.318 eV =   91289.0 cm**-1
F4dd                    =  0.259145554 a.u. =     7.052 eV =   56875.9 cm**-1
-------------------
Racah Parameters :
-------------------
B                       =  0.005550482 a.u. =     0.151 eV =    1218.2 cm**-1
C                       =  0.020567107 a.u. =     0.560 eV =    4514.0 cm**-1
C/B                     =  3.705
-------------------------------------------------------------------------------


----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV)  Energy(cm-1)      dz2        dxz        dyz        dx2-y2 ␣
↪   dxy
1          0.000        0.0        -0.927589   0.002893   0.009447   0.373452  -0.
↪003963
2          0.000        3.0         0.337599   0.005897   0.449370   0.827017  -0.
↪010128
3          0.000        3.0         0.160011  -0.005672  -0.893243   0.420094   0.
↪001383
4          0.001        4.4         0.000644   0.105816  -0.006612  -0.009602  -0.
↪994317
5          0.001        4.6         0.001541   0.994348  -0.007084  -0.002573   0.
↪105893
Ligand field orbitals were stored in ni.3d.nevpt2.lft.gbw
```

Note that:

- At the CASSCF level F0 (and subsequently racah A) is computed from CASSCF 2-electron coulomb integrals
- On the other hand at the NEVPT2 level F0 is not defined hence F0 and racah A are not printed. Below an alternative using the effective Slater exponnets will be provided.
- The LFT orbitals are saved in *.lft.gbw files which can be processed by the orca_plot to generate orbital visualization files.

AILFT provides a Fit quality analysis (see the original paper [433, 434])

Note: That at the CASSCF level the AI matrix of free atoms and ions is exactly parameterized in the chosen LFT parameterization scheme. As a result the RMS AI-LFT fitting errors is expected to be practically zero. This is not the case when a correlation treatment is chosen like NEVPT2 and the errors are expected to be somewhat larger.

The above is shown below:

```
Calculating statistical parameters                    ... done

Reference energy AI-LFT =    -38.134150221 au
Reference energy AI     =    -38.134150221 au

-----------------------------------------------
COMPARISON OF AB INITIO AND LIGAND FIELD RESULTS
-----------------------------------------------

Block   1
---------
AI-Root   0: E(AI)=    0.000 eV -> LF-Root   0:     0.000 eV    S= 0.998 Delta=   -
↪0.000 eV
AI-Root   1: E(AI)=    0.000 eV -> LF-Root   1:     0.000 eV    S= 0.981 Delta=   -
↪0.000 eV
AI-Root   2: E(AI)=    0.000 eV -> LF-Root   2:     0.000 eV    S= 0.980 Delta=   -
↪0.000 eV
AI-Root   3: E(AI)=    0.000 eV -> LF-Root   3:     0.000 eV    S= 0.773 Delta=   ␣
↪0.000 eV
AI-Root   4: E(AI)=    0.000 eV -> LF-Root   4:     0.000 eV    S= 0.774 Delta=   -
↪0.000 eV
AI-Root   5: E(AI)=    0.000 eV -> LF-Root   5:     0.000 eV    S= 0.985 Delta=   -
↪0.000 eV
AI-Root   6: E(AI)=    0.000 eV -> LF-Root   6:     0.000 eV    S= 0.986 Delta=   -
↪0.000 eV
AI-Root   7: E(AI)=    2.464 eV -> LF-Root   7:     2.464 eV    S= 0.998 Delta=   -
↪0.000 eV
AI-Root   8: E(AI)=    2.464 eV -> LF-Root   8:     2.464 eV    S= 0.998 Delta=   -
↪0.000 eV
AI-Root   9: E(AI)=    2.464 eV -> LF-Root   9:     2.464 eV    S= 1.000 Delta=   -
↪0.000 eV
RMS error for this block =     0.000 eV =       0.0 cm**-1

Block   2
---------
AI-Root   0: E(AI)=    2.033 eV -> LF-Root   0:     2.033 eV    S= 1.000 Delta=   -
↪0.000 eV
AI-Root   1: E(AI)=    2.033 eV -> LF-Root   1:     2.033 eV    S= 1.000 Delta=   -
↪0.000 eV
AI-Root   2: E(AI)=    2.033 eV -> LF-Root   2:     2.033 eV    S= 0.903 Delta=   -
↪0.000 eV
AI-Root   3: E(AI)=    2.033 eV -> LF-Root   3:     2.033 eV    S= 0.967 Delta=   -
↪0.000 eV
AI-Root   4: E(AI)=    2.033 eV -> LF-Root   4:     2.033 eV    S= 0.935 Delta=   -
↪0.000 eV
AI-Root   5: E(AI)=    3.183 eV -> LF-Root   5:     3.183 eV    S= 0.996 Delta=   -
↪0.000 eV
AI-Root   6: E(AI)=    3.183 eV -> LF-Root   6:     3.183 eV    S= 0.999 Delta=   -
↪0.000 eV
AI-Root   7: E(AI)=    3.183 eV -> LF-Root   7:     3.183 eV    S= 0.996 Delta=   -
↪0.000 eV
AI-Root   8: E(AI)=    3.183 eV -> LF-Root   8:     3.183 eV    S= 0.999 Delta=   -
↪0.000 eV
AI-Root   9: E(AI)=    3.183 eV -> LF-Root   9:     3.183 eV    S= 0.999 Delta=   -
↪0.000 eV
AI-Root  10: E(AI)=    3.183 eV -> LF-Root  10:     3.183 eV    S= 0.995 Delta=   -
↪0.000 eV
AI-Root  11: E(AI)=    3.183 eV -> LF-Root  11:     3.183 eV    S= 0.992 Delta=   -
↪0.000 eV
AI-Root  12: E(AI)=    3.183 eV -> LF-Root  12:     3.183 eV    S= 0.999 Delta=   -
↪0.000 eV
AI-Root  13: E(AI)=    3.183 eV -> LF-Root  13:     3.183 eV    S= 0.996 Delta=   -
```

(continues on next page)

(continued from previous page)

```
↪0.000 eV
AI-Root  14: E(AI)=    7.856 eV -> LF-Root  14:     7.856 eV    S= 1.000 Delta=   -
↪0.000 eV
RMS error for this block =     0.000 eV =       0.0 cm**-1

Total RMS error g=     0.000 eV =       0.0 cm**-1
Note: Dropped RMS error for the reference energy.

Confidence intervals (95) computed from the square root of the
diagonal elements of the covariance matrix:
H(dz2   ,dz2   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxz   ,dz2   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxz   ,dxz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dyz   ,dz2   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dyz   ,dxz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dyz   ,dyz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dx2-y2,dz2   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dx2-y2,dxz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dx2-y2,dyz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dx2-y2,dx2-y2)=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxy   ,dz2   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxy   ,dxz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxy   ,dyz   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxy   ,dx2-y2)=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
H(dxy   ,dxy   )=  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
F2dd            =  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
F4dd            =  0.000000000 a.u. =     0.000 eV =       0.0 cm**-1
Pearson's correlation coefficient =    1.000 (should be close to 1)
```

```
Calculating statistical parameters                    ... done

Reference energy AI-LFT =    -38.134345028 au
Reference energy AI     =    -38.134150221 au

-------------------------------------------------
COMPARISON OF AB INITIO AND LIGAND FIELD RESULTS
-------------------------------------------------

Block   1
---------
AI-Root   0: E(AI)=   -0.000 eV -> LF-Root   0:     0.000 eV    S= 0.933 Delta=   -
↪0.000 eV
AI-Root   1: E(AI)=    0.000 eV -> LF-Root   1:     0.000 eV    S= 0.769 Delta=   -
↪0.000 eV
AI-Root   2: E(AI)=    0.001 eV -> LF-Root   2:     0.000 eV    S= 0.773 Delta=   ␣
↪0.000 eV
AI-Root   3: E(AI)=    0.001 eV -> LF-Root   3:     0.000 eV    S= 0.742 Delta=   ␣
↪0.001 eV
AI-Root   4: E(AI)=    0.002 eV -> LF-Root   4:     0.000 eV    S= 0.750 Delta=   ␣
↪0.001 eV
AI-Root   5: E(AI)=    0.003 eV -> LF-Root   5:     0.001 eV    S= 0.931 Delta=   ␣
↪0.002 eV
AI-Root   6: E(AI)=    0.003 eV -> LF-Root   6:     0.001 eV    S= 0.998 Delta=   ␣
↪0.003 eV
AI-Root   7: E(AI)=    2.195 eV -> LF-Root   7:     2.266 eV    S= 1.000 Delta=   -
↪0.070 eV
AI-Root   8: E(AI)=    2.195 eV -> LF-Root   8:     2.266 eV    S= 0.998 Delta=   -
↪0.070 eV
AI-Root   9: E(AI)=    2.195 eV -> LF-Root   9:     2.266 eV    S= 0.998 Delta=   -
↪0.070 eV
RMS error for this block =     0.039 eV =     311.1 cm**-1
```

(continues on next page)

(continued from previous page)

```
Block   2
---------
AI-Root   0: E(AI)=    1.812 eV -> LF-Root   0:     1.875 eV    S= 0.825 Delta=   -
↪0.063 eV
AI-Root   1: E(AI)=    1.812 eV -> LF-Root   1:     1.875 eV    S= 0.938 Delta=   -
↪0.063 eV
AI-Root   2: E(AI)=    1.812 eV -> LF-Root   2:     1.875 eV    S= 1.000 Delta=   -
↪0.063 eV
AI-Root   3: E(AI)=    1.812 eV -> LF-Root   3:     1.875 eV    S= 1.000 Delta=   -
↪0.063 eV
AI-Root   4: E(AI)=    1.812 eV -> LF-Root   4:     1.875 eV    S= 0.773 Delta=   -
↪0.063 eV
AI-Root   5: E(AI)=    2.987 eV -> LF-Root   5:     2.932 eV    S= 0.955 Delta=   ␣
↪0.056 eV
AI-Root   6: E(AI)=    2.987 eV -> LF-Root   6:     2.932 eV    S= 0.910 Delta=   ␣
↪0.055 eV
AI-Root   7: E(AI)=    2.988 eV -> LF-Root   7:     2.932 eV    S= 0.874 Delta=   ␣
↪0.056 eV
AI-Root   8: E(AI)=    2.988 eV -> LF-Root   8:     2.932 eV    S= 0.792 Delta=   ␣
↪0.056 eV
AI-Root   9: E(AI)=    2.988 eV -> LF-Root   9:     2.932 eV    S= 0.796 Delta=   ␣
↪0.056 eV
AI-Root  10: E(AI)=    2.988 eV -> LF-Root  10:     2.932 eV    S= 0.808 Delta=   ␣
↪0.056 eV
AI-Root  11: E(AI)=    2.988 eV -> LF-Root  11:     2.932 eV    S= 0.971 Delta=   ␣
↪0.056 eV
AI-Root  12: E(AI)=    2.988 eV -> LF-Root  12:     2.932 eV    S= 0.994 Delta=   ␣
↪0.056 eV
AI-Root  13: E(AI)=    2.989 eV -> LF-Root  13:     2.932 eV    S= 0.994 Delta=   ␣
↪0.057 eV
AI-Root  14: E(AI)=    7.122 eV -> LF-Root  14:     7.241 eV    S= 1.000 Delta=   -
↪0.119 eV
RMS error for this block =     0.064 eV =     519.6 cm**-1

Total RMS error g=     0.057 eV =     457.2 cm**-1
Note: Dropped RMS error for the reference energy.

Confidence intervals (95) computed from the square root of the
diagonal elements of the covariance matrix:
H(dz2   ,dz2   )=  0.000523387 a.u. =     0.014 eV =     114.9 cm**-1
H(dxz   ,dz2   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dxz   ,dxz   )=  0.000523387 a.u. =     0.014 eV =     114.9 cm**-1
H(dyz   ,dz2   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dyz   ,dxz   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dyz   ,dyz   )=  0.000523387 a.u. =     0.014 eV =     114.9 cm**-1
H(dx2-y2,dz2   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dx2-y2,dxz   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dx2-y2,dyz   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dx2-y2,dx2-y2)=  0.000523387 a.u. =     0.014 eV =     114.9 cm**-1
H(dxy   ,dz2   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dxy   ,dxz   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dxy   ,dyz   )=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dxy   ,dx2-y2)=  0.000393965 a.u. =     0.011 eV =      86.5 cm**-1
H(dxy   ,dxy   )=  0.000523387 a.u. =     0.014 eV =     114.9 cm**-1
F2dd            =  0.002351095 a.u. =     0.064 eV =     516.0 cm**-1
F4dd            =  0.003038264 a.u. =     0.083 eV =     666.8 cm**-1
Pearson's correlation coefficient =    1.000 (should be close to 1)
```

Several utilities are offered for more specialized tasks that provide better control of the AILFT inputs and outputs:

- Skipping orbital otimization or reading in previously computed orbitals can be requested in two ways:

1. By the !NoIter keyword in the command line
2. By the AILFT_SkipOrbOpt in the ailft block (see example below)

- Estimating F0 SCPs or Racah A from single zeta Slater Exponents can be requested from the AILFT_EffectveSlaterExponents true keyword in the ailft block
- For the above task the knowledge of the principle quantum numbers is required. This can be specidied in two ways:
  1. By the AILFT_PQN x keyword in the ailft block (x=3 for 3d)
  2. By the LFTCase x keyword (LFTCase 3d, ommiting in this way the ActOrbs dOrbs keyword)

Let us see how all the above translates in the above example of the Ni^{2+} d^8 ion

```
!NoIter NEVPT2 def2-SVP def2-SVP/C

%casscf
 nel 8
 norb 5
 LFTCase 3d
 mult 3,1
 nroots 10,15
 ailft
  #AILFT_SkipOrbOpt true (An alternative to NoIter)
  #AILFT_PQNs 3 (Works together with ActOrbs dOrbs as an alternative to LFTCase 3d)
  AILFT_SlaterExponents true
 end
 rel
  dosoc true
 end
end

*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

By running the above input the fitted 1-electron energies and SCP parameters also Racah parameters for 1-shells will be printed at the CASSCF and NEVPT2 levels of theory, including F0s and Racah A as estimated from single zeta effective Slater exponents from the fitted F2dd SCPs.

```
------------------------------
AILFT MATRIX ELEMENTS (CASSCF)
--------------------------------

Ligand field one-electron matrix VLFT (a.u.) :
Orbital          dz2        dxz        dyz        dx2-y2     dxy
dz2         -7.974440   0.000000  -0.000000   0.000000  -0.000000
dxz          0.000000  -7.974440  -0.000000  -0.000000   0.000000
dyz         -0.000000  -0.000000  -7.974440  -0.000000   0.000000
dx2-y2       0.000000  -0.000000  -0.000000  -7.974440   0.000000
dxy         -0.000000   0.000000   0.000000   0.000000  -7.974440

--------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
--------------------------------------------------
--------------------------------------------------------------------------------
Computed Single Zeta Slater Effective Exponents                          ...
--------------------------------------------------------------------------------
kd(SZ)(from F2dd)       =  3.134434893 a.u.
--------------------------------------------------------------------------------
Computed F0s from Single Zeta Slater Effective Exponents                 ...
--------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
F0dd(from F2dd kd(SZ))  =  0.809116820 a.u. =    22.017 eV =  177580.6 cm**-1
F2dd                    =  0.427107567 a.u. =    11.622 eV =   93739.3 cm**-1
F4dd                    =  0.264174069 a.u. =     7.189 eV =   57979.5 cm**-1
-------------------------------------------------------------------------------
-------------------
Racah Parameters :
-------------------
A(from F2dd kd(SZ))     =  0.779764145 a.u. =    21.218 eV =  171138.4 cm**-1
B                       =  0.005721310 a.u. =     0.156 eV =    1255.7 cm**-1
C                       =  0.020966196 a.u. =     0.571 eV =    4601.5 cm**-1
C/B                     =  3.665
-------------------------------------------------------------------------------

----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV)  Energy(cm-1)      dz2        dxz        dyz        dx2-y2 ␣
↪   dxy
1          0.000        0.0        -0.999981   0.000787  -0.001735   0.004390  -0.
↪003810
2          0.000        0.0         0.003811   0.000493   0.000955   0.000497  -0.
↪999992
3          0.000        0.0         0.004388   0.000169   0.000080   0.999990   0.
↪000514
4          0.000        0.0        -0.000750  -0.999810  -0.019460   0.000174  -0.
↪000514
5          0.000        0.0        -0.001754  -0.019459   0.999809  -0.000069   0.
↪000939
Ligand field orbitals were stored in ni.3d.casscf.lft.gbw

...

------------------------------
AILFT MATRIX ELEMENTS (NEVPT2)
--------------------------------

Ligand field one-electron matrix VLFT (a.u.) :
Orbital          dz2        dxz        dyz        dx2-y2     dxy
dz2         -7.974812  -0.000000  -0.000000   0.000002  -0.000000
dxz         -0.000000  -7.974860   0.000000   0.000000   0.000000
dyz         -0.000000   0.000000  -7.974856  -0.000000   0.000000
dx2-y2       0.000002   0.000000  -0.000000  -7.974837   0.000000
dxy         -0.000000   0.000000   0.000000   0.000000  -7.974837

-------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
-------------------------------------------------
-------------------------------------------------------------------------------
Computed Single Zeta Slater Effective Exponents                          ...
-------------------------------------------------------------------------------
kd(SZ)(from F2dd)       =  3.126330433 a.u.
-------------------------------------------------------------------------------
Computed F0s from Single Zeta Slater Effective Exponents                 ...
-------------------------------------------------------------------------------
F0dd(from F2dd kd(SZ))  =  0.807024751 a.u. =    21.960 eV =  177121.5 cm**-1
F2dd                    =  0.426003229 a.u. =    11.592 eV =   93496.9 cm**-1
F4dd                    =  0.262001371 a.u. =     7.129 eV =   57502.7 cm**-1
-------------------------------------------------------------------------------
-------------------
Racah Parameters :
-------------------
```

(continues on next page)

(continued from previous page)

```
A(from F2dd kd(SZ))      =  0.777913487 a.u. =    21.168 eV =  170732.3 cm**-1
B                        =  0.005723406 a.u. =     0.156 eV =    1256.1 cm**-1
C                        =  0.020793760 a.u. =     0.566 eV =    4563.7 cm**-1
C/B                      =  3.633
-------------------------------------------------------------------------------


----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV)  Energy(cm-1)      dz2        dxz        dyz        dx2-y2 ␣
↪   dxy
1          0.000        0.0        -0.000608  -0.999795   0.017518   0.010019   0.
↪001244
2          0.000        0.9         0.000003  -0.017532  -0.999696  -0.003661   0.
↪016920
3          0.001        4.9         0.065092  -0.009854   0.004898  -0.995782   0.
↪063719
4          0.001        5.0        -0.004126   0.002173   0.016617   0.063640   0.
↪997824
5          0.001       10.5         0.997871   0.000042  -0.000237   0.065225  -0.
↪000030
Ligand field orbitals were stored in ni.3d.nevpt2.lft.gbw
```

It is also possible to treat only the High Spin states in the d- and f- 1-shell AILFT. Note that not all the cases can be treated as this renters the different SCP parameters undefined. In the beggining, AILFT will check whether such case is detected and will drop a warning message

For example in the case of the Fe^{2+} d^6 ion with an input like the following:

```
!NoIter def2-SVP def2-SVP/C


%casscf
 nel 6
 norb 5
 mult 5
 nroots 5
 LFTCase 3d
end

*xyz 2 3
Fe      0.0000000000      0.0000000000      0.0000000000
*
```

the following Warning will be printed in the beggining of the calculation

```
WARNING: In AILFT F2dd remains undefined when considering only the HS Multiplicity␣
↪block
Be Careful with the results!
TIP:     If possible use in addition a LS Multiplicity Block
```

Spin orbit coupling effects (SOC) can be introduced by parametrizing the effective SOC constant ζ. As long as SOC is requested in the rel CASSCF block the respective requested shell effective SOC constant ζ will be computed at the end of every AILFT calculation

Hence in the above examples one gets:

```
-----------------------------------------------
SPIN ORBIT COUPLING (based on CASSCF orbitals)
-----------------------------------------------
```

(continues on next page)

(continued from previous page)

```
AI-SOC-X integrals (cm-1)
0          1          2          3          4
0       0.000000   0.000000 1078.568273  -0.000000  -0.000000
1      -0.000000   0.000000  -0.000000  -0.000000 622.711683
2     -1078.568273   0.000000   0.000000 -622.711683   0.000000
3       0.000000   0.000000 622.711683   0.000000  -0.000000
4       0.000000 -622.711683  -0.000000   0.000000   0.000000
AI-SOC-Y integrals (cm-1)
0          1          2          3          4
0      -0.000000 -1078.568273  -0.000000   0.000000   0.000000
1     1078.568273   0.000000  -0.000000 -622.711683  -0.000000
2       0.000000   0.000000   0.000000  -0.000000 -622.711683
3      -0.000000 622.711683   0.000000   0.000000  -0.000000
4      -0.000000   0.000000 622.711683   0.000000   0.000000
AI-SOC-Z integrals (cm-1)
0          1          2          3          4
0       0.000000  -0.000000   0.000000  -0.000000   0.000000
1       0.000000  -0.000000 -622.711683  -0.000000   0.000000
2      -0.000000 622.711683   0.000000  -0.000000  -0.000000
3       0.000000   0.000000   0.000000  -0.000000 -1245.423365
4      -0.000000  -0.000000   0.000000 1245.423365   0.000000

Fit to the SOC matrix elements
a = 15.000000
b =     1.158 eV =    9340.7 cm**-1
SOC constant zeta =     0.077 eV =     622.7 cm**-1


LF-SOC-X integrals (cm-1)
0          1          2          3          4
0       0.000000  -0.000000 1078.568273  -0.000000  -0.000000
1       0.000000   0.000000  -0.000000  -0.000000 622.711683
2     -1078.568273   0.000000   0.000000 -622.711683  -0.000000
3       0.000000   0.000000 622.711683   0.000000  -0.000000
4       0.000000 -622.711683   0.000000   0.000000   0.000000
LF-SOC-Y integrals (cm-1)
0          1          2          3          4
0       0.000000 -1078.568273  -0.000000  -0.000000  -0.000000
1     1078.568273   0.000000  -0.000000 -622.711683  -0.000000
2       0.000000   0.000000   0.000000  -0.000000 -622.711683
3       0.000000 622.711683   0.000000   0.000000  -0.000000
4       0.000000   0.000000 622.711683   0.000000   0.000000
LF-SOC-Z integrals (cm-1)
0          1          2          3          4
0       0.000000  -0.000000  -0.000000  -0.000000  -0.000000
1       0.000000   0.000000 -622.711683  -0.000000  -0.000000
2       0.000000 622.711683   0.000000  -0.000000  -0.000000
3       0.000000   0.000000   0.000000   0.000000 -1245.423365
4       0.000000   0.000000   0.000000 1245.423365   0.000000

RMS error of nonzero matrix elements =    0.0 cm**-1


-----SOC-CONSTANTS-----
---All Values in cm-1---
ZETA_D = 622.71
------------------------
```

Starting fron ORCA 5.0 it is also possible in addition to CASSCF and NEVPT2 to employ DCDCAS(2) and Hermitian QD-NEVPT2 Abinitio Hamiltonians in AILFT Example inputs are provided below for DCDCAS(2):

```
!def2-SVP def2-SVP/C

%casscf
 nel 8
 norb 5
 actorbs dorbs
 mult 3,1
 nroots 10,15
 dcdcas true
 corrorder 2
 rel
  dosoc true
 end
end

*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

and Hermitian QD-NEVPT2:

```
!def2-SVP def2-SVP/C

%casscf
 nel 8
 norb 5
 actorbs dorbs
 mult 3,1
 nroots 10,15
 PTMethod  sc_nevpt2
 PTSettings
   QDType QD_VanVleck
 end
 rel
  dosoc true
 end
end

*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

Running the above inputs the respective DCDCAS(2) and Hermitian QD-NEVPT2 Hamiltonians will be processed:

```
Calculating ab initio Hamiltonian matrices               ...
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
DCDCAS Hamiltonian of block = 0 and order =  0 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
DCDCAS Hamiltonian of block = 0 and order =  1 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
DCDCAS Hamiltonian of block = 0 and order =  2 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
```

(continues on next page)

(continued from previous page)

```
DCDCAS Hamiltonian of block = 1 and order =  0 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
DCDCAS Hamiltonian of block = 1 and order =  1 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
DCDCAS correction for this block/order is calculated
DCDCAS Hamiltonian of block = 1 and order =  2 is passed
DCDCAS Hamiltonian diagonalized
------------------------------------------------------------

...

Calculating ab initio Hamiltonian matrices               ...
------------------------------------------------------------
Hermitian QD-NEVPT2 correction for this block is calculated
Hermitian QD-NEVPT2 Hamiltonian of block = 0 is passed
Hermitian QD-NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
Hermitian QD-NEVPT2 correction for this block is calculated
Hermitian QD-NEVPT2 Hamiltonian of block = 1 is passed
Hermitian QD-NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
```

It should be noted that NEVPT2 and Hermitian QD-NEVPT2 AILFT require a complete saturation of the excitation space. This implies that if less roots than the required are requested the AILFT analyis will be skipped in these cases. This is on the contrary not the case in CASSCF or DCDCAS(2) in which AILFT can operate under incomplete saturation of the excitation space.

```
Calculating ab initio Hamiltonian matrices               ...
WARNING: Number of NEVPT2 roots for block 0 (5) is not equal to the number of␣
↪CASCI CSFs (10)!
Skipping AILFT analysis with NEVPT2 energies!

WARNING: Number of NEVPT2 roots for block 1 (2) is not equal to the number of␣
↪CASCI CSFs (15)!
Skipping AILFT analysis with NEVPT2 energies!
```

In a similar fashion one can request a 2-shell AILFT calulation.

For this purpose the recomended steps are the following:

- In a first step the valence active space orbitals are optimized in the framework of SA-CASSCF calculation, e.g., the 3d MOs in a core 1s3d or 2p3d AILFT calculation, or the f MOs in an 4f5d AILFT calculation)
- In a second step the relevant core or virtual orbitals are rotated into the active space and the chosen CASCI/AILFT problem is solved by saturating the excitation space with all the involved excitations/multiplicity.
- In the most of the cases the excitation space of two multiplicities the High-Spin one and the subsequent Low-Spin one are enouph for a succesfull fitting of the parameters

It should be noted that 2-shell AILFT ivolves a 2-step fitting process following a bottom up shell angular momentum approach :

1. At first when possible an intra-shell fitting is performed
2. In following the respective effective Slater exponents are derived
3. In a last step an inter-shell fitting is performed and all the computed/fitted parameters are printed

This implies that:

- the flag of computing effective Slater exponents is always on by default in 2-shell AILFT
- the desired LFT problem is best requested by the LFTCase keywords (e.g. LFTCase 1s3d)

Let look at the case of 1s3d LFT problem of the Ni^{2+} d^8 ion. A relevant input is provided below:

```
!NoIter NEVPT2 def2-SVP def2-SVP/C

%method
 frozencore fc_none
end

%scf
 rotate
  {0,8,90}
 end
end

%casscf
 nel 10
 norb 6
 mult 3,1
 nroots 100,100
 LFTCase 1s3d
 rel
  dosoc true
 end
end


*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

Like in 1-shell AILFT, 2-shells AILFT starts with a sanity check

```
---- THE CAS-SCF GRADIENT HAS CONVERGED ----
--- FINALIZING ORBITALS ---
---- DOING ONE FINAL ITERATION FOR PRINTING ----
--- sd-orbitals (depends on the molecular axis frame)
L-Center:  0 Ni [0.000, 0.000, 0.000]
L-Center:  0 Ni Active Orbital 0 is the s orbital ; l = 0 ; active shell = 0
L-Center:  0 Ni Active Orbital 1 is one d orbital ; l = 2 ; active shell = 7
L-Center:  0 Ni Active Orbital 2 is one d orbital ; l = 2 ; active shell = 7
L-Center:  0 Ni Active Orbital 3 is one d orbital ; l = 2 ; active shell = 7
L-Center:  0 Ni Active Orbital 4 is one d orbital ; l = 2 ; active shell = 7
L-Center:  0 Ni Active Orbital 5 is one d orbital ; l = 2 ; active shell = 7
--- The active space contains 1 s orbitals and 5 d orbitals : OK
Setting 8 active MO to AO s       (0)
Setting 9 active MO to AO dz2     (11)
Setting 10 active MO to AO dxz     (12)
Setting 11 active MO to AO dyz     (13)
Setting 12 active MO to AO dx2y2   (14)
Setting 13 active MO to AO dxy     (15)
--- Canonicalize Internal Space
--- Canonicalize External Space
```

In following the AI-LFT Hamiltonians are constructed and the LFT parameters are fitted at the CASSCF and at the NEVPT2 levels of theory

```
-----------------------------
AILFT MATRIX ELEMENTS (CASSCF)
```

(continues on next page)

(continued from previous page)

```
--------------------------------

Ligand field one-electron matrix VLFT (a.u.) with V(0,0) fixed :
0          1          2          3          4          5
0     -334.652557  -0.000000   0.000000  -0.000000  -0.000000  -0.000000
1      -0.000000 -10.085777   0.000000   0.000000  -0.000000  -0.000000
2       0.000000   0.000000 -10.085777  -0.000000  -0.000000  -0.000000
3      -0.000000   0.000000  -0.000000 -10.085777   0.000000   0.000000
4      -0.000000  -0.000000  -0.000000   0.000000 -10.085777  -0.000000
5      -0.000000  -0.000000  -0.000000   0.000000  -0.000000 -10.085777

-------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
-------------------------------------------------
F0ss            = 17.137989284 a.u. =   466.348 eV = 3761353.9 cm**-1
F0dd            =  0.809116820 a.u. =    22.017 eV =  177580.6 cm**-1
F2dd            =  0.427107567 a.u. =    11.622 eV =   93739.3 cm**-1
F4dd            =  0.278548413 a.u. =     7.580 eV =   61134.3 cm**-1
F0sd            =  1.285327742 a.u. =    34.976 eV =  282096.8 cm**-1
G2sd            =  0.001928559 a.u. =     0.052 eV =     423.3 cm**-1
R2sddd          =  0.003748289 a.u. =     0.102 eV =     822.7 cm**-1

----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV) Energy(cm-1)         s          dz2        dxz        dyz  ␣
↪  dx2-y2       dxz
1          0.000        0.0          -1.000000  -0.000000   0.000000  -0.000000  -
↪0.000000   0.000000
2       8831.911  71234174.2           0.000000  -0.006282   0.006583  -0.271975  ␣
↪0.962261   0.000000
3       8831.911  71234174.2           0.000000   0.000000   0.000000   0.000000  ␣
↪0.000000   1.000000
4       8831.911  71234174.2           0.000000  -0.075508  -0.016185  -0.959327  -
↪0.271528   0.000000
5       8831.911  71234174.2          -0.000000   0.071391  -0.997365   0.008467  ␣
↪0.009682   0.000000
6       8831.911  71234174.2          -0.000000   0.994566   0.070404  -0.075159  -
↪0.015232  -0.000000
Ligand field orbitals were stored ni.1s3d.casscf.lft.gbw

...

------------------------------
AILFT MATRIX ELEMENTS (NEVPT2)
--------------------------------

Ligand field one-electron matrix VLFT (a.u.) with V(0,0) fixed :
0          1          2          3          4          5
0     -334.652557  -0.000000   0.000000   0.000000   0.000000   0.000000
1      -0.000000 -10.298799   0.000008   0.000006  -0.000002  -0.000002
2       0.000000   0.000008 -10.293700  -0.000042  -0.000008  -0.000006
3       0.000000   0.000006  -0.000042 -10.293839  -0.000001   0.000008
4       0.000000  -0.000002  -0.000008  -0.000001 -10.293927  -0.000033
5       0.000000  -0.000002  -0.000006   0.000008  -0.000033 -10.293511

-------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
-------------------------------------------------
F0ss            = 16.677683435 a.u. =   453.823 eV = 3660328.4 cm**-1
F0dd            =  0.806859685 a.u. =    21.956 eV =  177085.2 cm**-1
```

(continues on next page)

(continued from previous page)

```
F2dd            =  0.425916096 a.u. =    11.590 eV =   93477.8 cm**-1
F4dd            =  0.277771367 a.u. =     7.559 eV =   60963.8 cm**-1
F0sd            =  1.424742585 a.u. =    38.769 eV =  312694.9 cm**-1
G2sd            =  0.025339297 a.u. =     0.690 eV =    5561.3 cm**-1
R2sddd          =  0.003739506 a.u. =     0.102 eV =     820.7 cm**-1

----------------------------------------
The ligand field one electron eigenfunctions:
----------------------------------------
Orbital    Energy (eV) Energy(cm-1)         s          dz2        dxz        dyz  ␣
↪  dx2-y2       dxz
1          0.000        0.0           1.000000   0.000000  -0.000000  -0.000000  -
↪0.000000  -0.000000
2       8826.114  71187421.2          -0.000000   0.999998  -0.001607  -0.001229  ␣
↪0.000405   0.000329
3       8826.247  71188489.9          -0.000000   0.000330  -0.040203  -0.027545  -
↪0.995773  -0.077852
4       8826.249  71188507.4          -0.000000   0.001631   0.264542   0.963492  -
↪0.035726  -0.020544
5       8826.254  71188542.9           0.000000  -0.001223  -0.962932   0.264866  ␣
↪0.034492  -0.037635
6       8826.258  71188582.5           0.000000  -0.000317  -0.034070   0.027728  -
↪0.077265   0.996042
Ligand field orbitals were stored in ni.1s3d.nevpt2.lft.gbw
```

As discussed above saturation of the excitation space is a requirement also in the case of 2-shell AILFT. It is usually enough to specify a large number of roots for two multiplicites (e.g. 100 singlets and triplets in the above example) The exact number of roots will be automatically detected.

```
Multiplicity                        ...    3
#(Configurations)                   ...   15
#(CSFs)                             ...   15
#(Roots)                            ...  100
WARNING (ORCA_CASSCF): NRoots > NCSFs. Adjusting to maximum number of roots.␣
↪Please check the output carefully!
#(Roots)                            ...   15
ROOT=0 WEIGHT=    0.033333
ROOT=1 WEIGHT=    0.033333
ROOT=2 WEIGHT=    0.033333
ROOT=3 WEIGHT=    0.033333
ROOT=4 WEIGHT=    0.033333
ROOT=5 WEIGHT=    0.033333
ROOT=6 WEIGHT=    0.033333
ROOT=7 WEIGHT=    0.033333
ROOT=8 WEIGHT=    0.033333
ROOT=9 WEIGHT=    0.033333
ROOT=10 WEIGHT=    0.033333
ROOT=11 WEIGHT=    0.033333
ROOT=12 WEIGHT=    0.033333
ROOT=13 WEIGHT=    0.033333
ROOT=14 WEIGHT=    0.033333

BLOCK  2 WEIGHT=   0.5000
Multiplicity                        ...    1
#(Configurations)                   ...   21
#(CSFs)                             ...   21
#(Roots)                            ...  100
WARNING (ORCA_CASSCF): NRoots > NCSFs. Adjusting to maximum number of roots.␣
↪Please check the output carefully!
#(Roots)                            ...   21
ROOT=0 WEIGHT=    0.023810
```

(continues on next page)

(continued from previous page)

```
ROOT=1 WEIGHT=    0.023810
ROOT=2 WEIGHT=    0.023810
ROOT=3 WEIGHT=    0.023810
ROOT=4 WEIGHT=    0.023810
ROOT=5 WEIGHT=    0.023810
ROOT=6 WEIGHT=    0.023810
ROOT=7 WEIGHT=    0.023810
ROOT=8 WEIGHT=    0.023810
ROOT=9 WEIGHT=    0.023810
ROOT=10 WEIGHT=    0.023810
ROOT=11 WEIGHT=    0.023810
ROOT=12 WEIGHT=    0.023810
ROOT=13 WEIGHT=    0.023810
ROOT=14 WEIGHT=    0.023810
ROOT=15 WEIGHT=    0.023810
ROOT=16 WEIGHT=    0.023810
ROOT=17 WEIGHT=    0.023810
ROOT=18 WEIGHT=    0.023810
ROOT=19 WEIGHT=    0.023810
ROOT=20 WEIGHT=    0.023810
```

However very often the required number of states to be computed in the framework of NEVPT2 type of calculations are quite large. In these cases a Hamiltonian reduction process on the basis of the Restrictive Active Space (RAS) is required. In fact all LFT parameters can be determined by considering up to double excitations from the donor-shell.

Let us consider the 2p3d case of the Fe^{2+} d^6 ion. Saturation of the active space requires to consider 70 triplet and 378 singlet states. Restriction of the active space to only up to double excitations from the 2p-shell results in 65 quintet and 330 triplet states. The Hamiltonian reduction can be requested in the ailft block:

```
ailft
 AILFT_Dimension 2 # Up to double excitations from the donor shell
                 Other options:
                 1   Up to single excitations from the donor shell
                 0   No excitations from the donor shell
end
```

Hence the relevant input can be now formulated as:

```
!NoIter MOREAD DKH2 DKH-def2-TZVP def2-TZVP/C NEVPT2

%moinp "fe.3d.gbw"

%pal
 nprocs 16
end

%method
 frozencore fc_none
end

%scf
 rotate
  {2,6,90}
  {3,7,90}
  {4,8,90}
 end
end

%casscf
 nel 12
 norb 8
```

(continues on next page)

(continued from previous page)

```
 mult 5,3
 nroots 65,330
 LFTCase 2p3d
 ailft
  AILFT_Dimension 2
 end
 rel
  dosoc true
  GTensor false
  DoDTensor false
 end
end

*xyz 2 5
Fe      0.0000000000      0.0000000000      0.0000000000
*
```

Note that before running the above calculation:

- An initial SA-CASSCF calculation has been performed on the valence states of Fe in the 3d active space. These orbitals (fe.3d.gbw) are read in
- The computation of the g- and D- tensors is switched off. This is recomended if the magnetism analysis is not required
- As core spectroscopy is targeted the frozen core is switched off

At the NEVPT2 part the reduced AI and LFT Hamiltonians will be constructed

```
Calculating  ab initio Hamiltonian matrices              ...
------------------------------------------------------------
NRoots (NEVPT2) for this block = 65
NEVPT2 correction for this block is calculated
Full NEVPT2 Hamiltonian constructed
Full NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
------------------------------------------------------------
NRoots (NEVPT2) for this block = 330
NEVPT2 correction for this block is calculated
Full NEVPT2 Hamiltonian constructed
Full NEVPT2 Hamiltonian diagonalized
------------------------------------------------------------
```

As a result the CASSCF and NEVPT2 LFT parameters will be determined in the requested reduced basis

```
------------------------------
AILFT MATRIX ELEMENTS (CASSCF)
--------------------------------


--------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
--------------------------------------------------
F0pp            =  3.889665545 a.u. =   105.843 eV =  853682.9 cm**-1
F2pp            =  1.832901835 a.u. =    49.876 eV =  402275.5 cm**-1
F0dd            =  0.767706319 a.u. =    20.890 eV =  168492.1 cm**-1
F2dd            =  0.405248254 a.u. =    11.027 eV =   88941.7 cm**-1
F4dd            =  0.264292339 a.u. =     7.192 eV =   58005.5 cm**-1
F0pd            =  1.174187892 a.u. =    31.951 eV =  257704.5 cm**-1
F2pd            =  0.220959621 a.u. =     6.013 eV =   48495.0 cm**-1
G1pd            =  0.157243007 a.u. =     4.279 eV =   34510.9 cm**-1
G3pd            =  0.089473762 a.u. =     2.435 eV =   19637.2 cm**-1
```

(continues on next page)

(continued from previous page)

```
...

-----------------------------
AILFT MATRIX ELEMENTS (NEVPT2)
------------------------------

--------------------------------------------------
Slater-Condon Parameters (electronic repulsion) :
--------------------------------------------------
F0pp            =  3.527444471 a.u. =    95.987 eV =  774184.6 cm**-1
F2pp            =  1.906123325 a.u. =    51.868 eV =  418345.7 cm**-1
F0dd            =  0.808248242 a.u. =    21.994 eV =  177390.0 cm**-1
F2dd            =  0.426649072 a.u. =    11.610 eV =   93638.6 cm**-1
F4dd            =  0.278249395 a.u. =     7.572 eV =   61068.7 cm**-1
F0pd            =  1.657417509 a.u. =    45.101 eV =  363761.1 cm**-1
F2pd            =  0.198646995 a.u. =     5.405 eV =   43598.0 cm**-1
G1pd            =  0.206111579 a.u. =     5.609 eV =   45236.3 cm**-1
G3pd            =  0.128174221 a.u. =     3.488 eV =   28131.0 cm**-1

...
```

In the above example inclusion of SOC will result in the computation of the effective SOC ζ constants of both p and d shells:

```
-----SOC-CONSTANTS-----
---All Values in cm-1---
ZETA_P = 65018.19
ZETA_D = 453.53
------------------------
```

One important feature of 1- and in particular of the 2-shell AILFT is that it is connected to the standalone orca_lft multiplet program. Hence every succesfull AILFT calculation will automatically construct relevant inputs for the orca_lft.

For example in the avove 2p3d case of the Fe^{2+} d^6 ion the following inputs will be constructed

```
fe.2p3d.casscf.lft.inp
fe.2p3d.nevpt2.lft.inp
```

with the NEVPT2 one looking like this:

```
%lft

#-----Parameters------
NEl= 12
Shell_PQN= 0, 2, 3, 0
Mult= 5, 3
NRoots= 65, 330
#--------------------


#---Slater-Condon Parameters---
#---All Values in eV---
PARAMETERS
F0pp =  95.9866
F2pp =  51.8683
F0dd =  21.9936
F2dd =  11.6097
F4dd =  7.5716
F0pd =  45.1006
F2pd =  5.4055
```

(continues on next page)

(continued from previous page)

```
G1pd =  5.6086
G3pd =  3.4878
end
#--------------------


#---LFT-Matrix Elelemnts---
#---All Values in eV---
FUNCTIONS
0   0   "  0.0000"
1   0   " -0.0146"
1   1   "  0.0947"
2   0   "  0.0210"
2   1   "  0.0061"
2   2   "  0.1155"
3   0   "  0.0000"
3   1   " -0.0000"
3   2   " -0.0000"
3   3   "1086.2398"
4   0   "  0.0000"
4   1   " -0.0000"
4   2   " -0.0000"
4   3   "  0.0323"
4   4   "1086.2181"
5   0   " -0.0000"
5   1   "  0.0000"
5   2   "  0.0000"
5   3   " -0.0356"
5   4   " -0.0154"
5   5   "1086.1183"
6   0   "  0.0000"
6   1   " -0.0000"
6   2   " -0.0000"
6   3   " -0.0767"
6   4   " -0.0080"
6   5   "  0.0341"
6   6   "1086.1219"
7   0   " -0.0000"
7   1   "  0.0000"
7   2   "  0.0000"
7   3   "  0.0050"
7   4   " -0.0023"
7   5   "  0.0026"
7   6   " -0.0038"
7   7   "1086.0688"
end
#--------------------


#---SOC-CONSTANTS---
#---All Values in eV---
PARAMETERS
ZETA_P = 8.06
ZETA_D = 0.06
end
#--------------------


#---SPECTRA/PROPERTIES---
DoABS true
#------------------------
```

(continues on next page)

(continued from previous page)

```
end


*xyz Charge Multiplicity
Atom 0.00 0.00 0.00
*
```

Further details regardin orca_lft can be found in the orca_lft section Section 9.2.15 and the orca_lft tutorial. (*orca_lft*) and the orca_lft tutorial.

3.13.17 Extended Space Ab initio Ligand Field Theory (ESAILFT)

While the one- and two-shell AILFT approach has been remarkably effective in extracting interpretable parameters. There are still limitation to this approach. The nature of the AILFT extraction requires the CASCI or effective Hamiltonian matrix to be of the same dimension as the LFT Hamiltonian. This means that an extension of the CASSCF calculation, with the active space larger than the d-space to incorporate static correlations without increasing the number of AILFT parameters, is not possible with the original recipes.

We now provide a solution to this limitation by introducing another effective Hamiltonian that is based on partitioning the extended active spaces in CASSCF calculations. This method (esAILFT)[456] allows us to perform AILFT calculations on the basis of extended CASSCF active spaces to, in principle, any number of orbitals. The central idea is that esAILFT would allow for the inclusion of effects such as radial correlation of d-orbitals and reduce the overestimation of the ionic character in the CASSCF calculation.

The method of partitioning given by Löwdin presents one approach for building effective Hamiltonians. In this case, the Hilbert space is divided into a model (A) and an outer (B) space. The time-independent Schrödinger equation can be rewritten as:

$$\mathbf{H}^{AI}\begin{pmatrix}\mathbf{C}_A\\ \mathbf{C}_B\end{pmatrix} = \begin{pmatrix}\mathbf{H}_{AA} & \mathbf{H}_{AB}\\ \mathbf{H}_{BA} & \mathbf{H}_{BB}\end{pmatrix}\begin{pmatrix}\mathbf{C}_A\\ \mathbf{C}_B\end{pmatrix} = E^{(0)}\cdot\begin{pmatrix}\mathbf{C}_A\\ \mathbf{C}_B\end{pmatrix}$$

$$\mathbf{H}^{AI}_{part} = \mathbf{H}_{AA} - \mathbf{H}_{AB}(\mathbf{H}_{BB} - 1\cdot E_0)^{-1}\mathbf{H}_{BA}$$

Here E_0 can be fixed to the lowest energy of the CASCI matrix but a bias-correction term becomes necessary to correct the energies of $\mathbf{H}^{AI}_{part}$

$$E_I^{BC} = [E_I - E_0]\times\sum_{k\in B}\frac{H^{AI}_{IK}H^{AI}_{KI}}{(E_K - E_0)^2}$$

The individual steps in the calculation are summarized as follows:

1. Orbital preparation Optimized CASSCF MOs are obtained by solving the CASSCF equations. We then exploit the unitary invariance of the active orbital space in order to make these orbitals suitable for the extended AILFT procedure by performing the following three steps
    1. The active orbitals are localized which will separate metal-based d- or from ligand-based orbitals
    2. The metal-d- orbitals are canonicalized by diagonalizing the matrix representation of the angular momentum operator $\hat{L}$ z and then phase-adjusted to be consistent with the ligand field orbitals. Both steps 1.1. and 1.2. are already present in the original AILFT procedure.
    3. The remaining outer space orbitals are transformed to diagonalize the corresponding sub-block of the Fock operator. This step is optional but, in our experience, beneficial for the partitioning procedure and the interpretation of the results.
2. Generation of the effective Hamiltonian over the ligand field manifold space. The individual steps are as follows
    1. Calculations of the actual effective Hamiltonian
    2. Calculation of the bias correction
3. The original AILFT procedure is then used on the effective Hamiltonian.

We can illustrate this with an example for V^{2+}. The process begins with a converged minimal space CASSCF calcutions.

```
!x2c x2c-TZVPall AUTOAUX
%maxcore 4000
%casscf
  # run the AILFT with the minimal active space
  actorbs dorbs
  # definition of the active space, multiplicity and number of roots
  nel 3
  norb 5
  mult 4,2
  nroots 10,40
  trafostep RI
  # optional properties such as SOC splitting and g-matrix
  rel
    dosoc false
    gtensor false
  end
end
# geometry block
*xyz 2 4
  V 0.0 0.0 0.0
*
# enable picture change
%rel
  picturechange true
end
```

We then run an extended space CASSCF:

```
!x2c x2c-TZVPall AUTOAUX moread
%moinp "V2p_min.gbw" # contains the converged orbitals with the minimal active␣
↪space
%maxcore 4000
%casscf
  # definition of the extended active space
  norb 9
  nel 11
  mult 4,2
  nroots 10,40
  trafostep RI
end
*xyz 2 4
  V 0.0 0.0 0.0
*
# enable picture change
%rel
  picturechange true
end
```

This can then be used for computing LFT parameters:

```
!x2c x2c-TZVPall AUTOAUX moread
%moinp "V2p_3s3p.gbw" # converged casscf orbitals with the extended active space
%maxcore 4000
%casscf
  # run the esAILFT
  actorbs lmorbs
  ailft
   AILFT_SkipOrbOpt true
   NOrbInternal 4 # no of orbitals in the extended space to be labelled as internal
```

(continues on next page)

(continued from previous page)

```
   NOrbExternal 0 # no of orbitals in the extended space to be labelled as external
   MinElectrons 3 # no of electrons in the active space after external + internal␣
↪are cut
  end
  # definition of the active space, multiplicity and number of roots
  norb 9
  nel 11
  mult 4,2
  nroots 10,40
  trafostep RI
  rel
    dosoc false
    gtensor false
  end
end
*xyz 2 4
  V 0.0 0.0 0.0
*
%rel
  picturechange true
end
```

The output will include a block dedicated for ESAILFT. The information displayed is in an identical format to the one-shell case except it is extracted from the partitioned hamiltonian.

```
------------------------------
AILFT MATRIX ELEMENTS (Hermitian HEffective)
--------------------------------
```

3.13.18 Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS

Starting from ORCA 4.1, a CASCI/NEVPT2 protocol can be used to compute core excited spectra, namely X-ray absorption (XAS) and resonant inelastic scattering (RIXS) spectra. RASCI calculations can also be easily specified.

The XAS/RIXS spectra calculations requires two steps:

- In a first step one needs to optimize the valence active space orbitals in the framework of SA-CASSCF calculations, e.g. including valence excited states in the range between 6 to 15 eV.
- In a second step the relevant core orbitals are rotated into the active space and the (C/R)ASCI/NEVPT2 problem is solved by saturating the excitation space with singly core-excited electronic configurations using the previously optimized sets of orbitals

Further information can be found in reference[457]

A relevant input for Fe L-edge XAS calculation of a Fe(III) complex like Fe(acac)$_3$ is given below for CASCI/NEVPT2:

```
%moinp "Fe_acac3_casscf.gbw"

%scf
rotate
{4,89,90,0,0}
{3,88,90,0,0}
{2,87,90,0,0}
end
end

%rel
picturechange true
FiniteNuc true
```

(continues on next page)

(continued from previous page)

```
end

%method FrozenCore FC_NONE
end

# CASSCF/NEVPT2 on the valence and L-edge excited states
%casscf
nel 11
norb 8
mult 6,4
nroots 16,173
maxiter 1
# account for spin-orbit coupling
rel
  DoSOC true
end
# adding the dynamical correlation with NEVPT2
PTMethod SC\_NEVPT2
end

* xyz 0 6
...
*
```

For RASCI/NEVPT2 calculations the valence d AS is set to RAS2. The RAS3 space is usually set empty. The RAS1 space contains the previously rotated core orbitals. To generate a single core hole, the number of maximum holes in the RAS1 space must be set to 1. Accordingly, the maximum number of particles in the RAS3 space must be 0. The RASCI input should thus look the following

```
%casscf
 nel 11
 norb 8
 ...
 refs
  ras(11:3 1/5/0 0) # (Nel: NRAS1 MaxHoles / NRAS2 / NRAS3 MaxParticles)
 end
 ...
end
```

As it is explicitly described in the respective ROCIS section RIXS spectra can be requested by the following keywords:

```
RIXS    true   # Request RIXS calculation (NoSOC)
RIXSSOC true   # Request RIXS calculation (with SOC)
Elastic true   # Request RIXS calculation (Elastic)
```

Please consult section Section 5.7.4 for processing and analyzing the generated spectra.

3.13.19 Core Excited Spectra: CAS-CI/RAS-CI XES

Starting from ORCA 5.0 likewise to RASCI-XES (see section Section 3.19.14) orca features a CASCI-XES protocol.

Likewise to the RASCI-XES the CASCI-XES calculations requires two steps:

- In a first step one needs to optimize the valence active space orbitals in the framework of SA-CASSCF calculations, e.g. including valence excited states in the range between 6 to 15 eV.
- In a second step the relevant core orbitals e.g metal 1s and 3p are rotated into the active space and the CASCI problem is solved for the ionized system by saturating the excitation space with singly core-excited electronic configurations using the previously optimized sets of orbitals. In CASCI-XES this can be acheived by defining reference configurations. or via the RAS functionality.

The XESSOC calculation is called by speciyfing the following keywords in the rel block:

```
rel
XESSOC true
XASMOs Number of the rotated 1s MO
end
```

Following a SA-CASSCF calculation:

```
! X2c x2c-TZVPall x2c/J  def2-TZVP/C
! NormalPrint
! NoLoewdin NoMulliken

%casscf
nel 5
norb 5
nroots 1, 24
mult 6, 4
end


* xyz -3 6
Fe      0       0       0
Cl      2.40    0       0
Cl      -2.40   0       0
Cl      0       2.40    0
Cl      0       -2.40   0
Cl      0       0       2.40
Cl      0       0       -2.40
*
```

A relevant input for Fe XES calculation of a Fe(III) complex like $FeCl_6$ is given below:

```
! def2-SVP def2-SVP/C ZORA CPCM PAL8
! NormalPrint
! MOREAD
! NoLoewdin NoMulliken


%moinp "FeCl6_casscf.gbw"

%scf
#Rotate the 1s and 3p orbitals below the SOMOs by using the rotate option
rotate {0,59,90} {36,60,90} {37,61,90} {38,62,90} end
end

%casscf
nel 12
norb 9
nroots 1000, 1000
mult 7, 5
maxiter 1
refs
  1 2 2 2 0 0 1 2 2
  1 2 2 2 0 0 2 1 2
  1 2 2 2 0 0 2 2 1
  1 2 2 2 0 1 0 2 2
  1 2 2 2 0 1 1 1 2
  1 2 2 2 0 1 1 2 1
  1 2 2 2 0 1 2 0 2
...
  2 2 2 2 2 0 0 2 0
  2 2 2 2 2 0 1 0 1
```

(continues on next page)

(continued from previous page)

```
  2 2 2 2 2 0 1 1 0
  2 2 2 2 2 0 2 0 0
  2 2 2 2 2 1 0 0 1
  2 2 2 2 2 1 0 1 0
  2 2 2 2 2 1 1 0 0
  2 2 2 2 2 2 0 0 0
end
DoDipoleVelocity true
DecomposeFosc true
rel
dosoc true
XESSOC true
XASMOs 59
DoDTensor false
DoVelocity true
end
end


%method
SpecialGridAtoms 26 #Increase the radial integration accuracy on Fe
SpecialGridIntAcc 7 #Requested radial integration accuracy values
end

* xyz -2 5
Fe      0       0       0
Cl      2.40    0       0
Cl      -2.40   0       0
Cl      0       2.40    0
Cl      0       -2.40   0
Cl      0       0       2.40
Cl      0       0       -2.40
*
```

In ORCA 6 all these type of inputs have been unified across the wavefunction based modiles (e.g. CASSCF MRCI, RASCI, ROCIS and LFT) so in a similar fashion to *X-ray Spectroscopy* one can also specify the respective RAS excitation space as following:

```
%casscf
nel 12
norb 9
nroots 1000, 1000
mult 7, 5
maxiter 1
refs ras(12:4 1/5/ 0 0) end
DoDipoleVelocity true
DecomposeFosc true
rel
  DoSOc true
  Xessoc true
  xasmos 59
  DoDTensor false
  DoVelocity true
 end
end
```

In the above inputs one notes that the exact knowledge of the states to saturate the excitations space is not required. One only need to specify a large number (e.g. 1000) and the program will automatically detect the required 19 septet and 270 quintet states:

```
BLOCK  1 WEIGHT=   0.5000
Multiplicity                        ...    7
#(Configurations)                   ...    4
#(CSFs)                             ...    4
#(Roots)                            ... 1000
WARNING (ORCA_CASSCF): NRoots > NCSFs. Adjusting to maximum number of roots.␣
↪Please check the output carefully!
#(Roots)                            ...    4

...

BLOCK  2 WEIGHT=   0.5000
Multiplicity                        ...    5
#(Configurations)                   ...   89
#(CSFs)                             ...  105
#(Roots)                            ... 1000
WARNING (ORCA_CASSCF): NRoots > NCSFs. Adjusting to maximum number of roots.␣
↪Please check the output carefully!
#(Roots)                            ...  105
```

By now running the above input for the 4 septet and the 81 quintet states the following output is generated

```
-----------------------------------------------------------------------------------
↪---------------------
      SOC CORRECTED   EMISSION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪---------------------
      Transition          Energy      Energy  Wavelength fosc(D2)       D2        |DX|␣
↪      |DY|       |DZ|
                           (eV)       (cm-1)     (nm)                  (au**2)     (au)␣
↪      (au)       (au)
-----------------------------------------------------------------------------------
↪---------------------
 421-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000000870   0.00000  ␣
↪0.00000   0.00000   0.00007
 423-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000001088   0.00000  ␣
↪0.00006   0.00006   0.00000
 424-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000000898   0.00000  ␣
↪0.00002   0.00003   0.00006
 422-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
 426-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000001048   0.00000  ␣
↪0.00005   0.00005   0.00003
 425-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000000664   0.00000  ␣
↪0.00005   0.00004   0.00001
 427-7.0A  -> 420-5.0A   7084.189573 57137848.2     0.2   0.000000653   0.00000  ␣
↪0.00004   0.00004   0.00000
 421-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000001233   0.00000  ␣
↪0.00007   0.00005   0.00003
 423-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000000354   0.00000  ␣
↪0.00002   0.00002   0.00004
 422-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000001427   0.00000  ␣
↪0.00004   0.00008   0.00000
 424-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000000467   0.00000  ␣
↪0.00005   0.00002   0.00001
 426-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000000216   0.00000  ␣
↪0.00003   0.00002   0.00001
 425-7.0A  -> 419-5.0A   7084.189691 57137849.2     0.2   0.000001334   0.00000  ␣
↪0.00001   0.00002   0.00009
 421-7.0A  -> 418-5.0A   7084.189691 57137849.2     0.2   0.000000757   0.00000  ␣
↪0.00000   0.00005   0.00004
...
```

Finally by processing the .out file with orca_mapspc:

```
orca_mapspc fecl6_xes.out XESSOC -x07000 -x17200 -w4.0 -eV -n10000
```

and by plotting the resulted XES spectrum one will get the respective RASCI-XES spectrum presented in Fig. 3.23

This will result in Fig. 3.23.

Fig. 3.23: Calculated RASCI K_β Mainline XES spectrum of $[Fe(Cl)_6]^{3+}$

Since Orca 6.0 the computed transition moments in the presence of SOC can be taken beyond the dipole approximation by using the OPS tool, check section Section 5.14 for details.

All the possible approximations can be requested with the following commands:

```
#Non-Relativistic/Relativistic treatment

%casscf
 DoDipoleLength        true
 DoDipoleVelocity      true
 DecomposeFosc         true
 DoFullSemiclassical   true
end
```

Note

The recomended process for generating XES spectra

- within the electric dipole approximation (EDA) is via `DoDipoleLength` or `DoDipoleVelocity`

- beyond EDA, employing the Full Field Matter Interatcion Operator FFMIO via the `DoFullSemiclassical`

Orca_mapspc can process all these files. The list of the relevant keywords in the case of RASCI-XES protocol is

```
XESSOC  #This will process the SOC corrected beyond the electric dipole␣
↪approcimation (EDA)
XESSOCFFMIO #This will process the SOC corrected FFMIO X-ray Emission spectra
```

A more complete list can be found in (*orca_mapspc*)

Note

This is inpriciple the recomended protocol for computing X-ray related spectra XAS, XES, RXES, RIXS:

- As it provides more direct and computationally efficient protocol for computing
- Metal K, L and M edge XAS spectra
- K_α: Fe $2p \rightarrow$ Fe $1s$
- $K_{\beta_{1,3}}$: Fe $3p \rightarrow$ Fe $1s$ (Mainline XES)
- $K_{\beta_{2,5}}$ / $K_{\beta'}$: (Cl $2p$ + Fe $3d$) $\rightarrow$ Fe $1s$ (VtC-XES)
- as well the respective resonant XES (RIXS) spectra.
- Combined with NEVPT2 the protocol can actually treat quite challenging cases in terms of electron correltion.
- We should mention that within the limits of their applicability XES and RIXS spectra can also be computed on the basis of Static Ground State DFT (SGS-DFT), see discussion in *Static Ground State DFT (SGS-DFT)* and ROCIS family of methods, see discussion in *Resonant Inelastic Scattering Spectroscopy*.
- A general discussion regarding X-ray spectroscopy experiements and how to approach them on the basis of modern computational spectroscopy methodologies is provided in reference [458]

3.14 Approximate Full CI Calculations in Subspace: ICE-CI

3.14.1 Introduction

In many circumstances, one would like to generate a wavefunction that is as close as possible to the full-CI result, but Full CI itself is out of the question for computational reasons. Situations in which that may be desirable include a) one wants to generate highly accurate energies for small molecules or b) one wants to sort out a number of low-lying states or c) one wants to run *CASSCF calculations* with larger active spaces than the about fourteen orbitals that have been the state of the art for a long time.

ORCA features a method that has been termed Iterative-Configuration Expansion Configuration Interaction (ICE-CI).[443, 444] It is based on much older ideas brought forward by Jean-Paul Malrieu and his co-workers in the framework of the CIPSI (an abbreviation for a method with a rather bulky name *Configuration Interaction by Perturbation with multiconfigurational zeroth-order wave functions Selected by Iterative process*) in the early 1970s.

The goal of the ICE-CI is to provide compact wavefunction(s) (e.g. one or several states) close to the full-CI limit at a small fraction of the computational cost. However, ICE-CI itself is not designed to deal with hundreds of atoms or thousands of basis functions. Thus, unlike, say DLPNO-CCSD(T) which is a high accuracy method for treating large sytems, ICE-CI is either a highly robust high accuracy method for very small systems or a "building block" for large systems. By itself it can treat a few dozen electrons and orbitals – e.g. *much* more than full CI – but it cannot

do wonders. Its scope is similar to the density matrix renormalization group (DMRG) or Quantum Monte Carlo Full CI (QMCFCI) procedures.

ICE-CI should be viewed as a multireference approach. It is self-adaptive and robust, even in the presence of near or perfect degeneracies. It yields orthogonal states (when applied to several states) and spin eigenfunctions. It also yields a density and a spin density.

3.14.2 The ICE-CI and CIPSI Algorithms

The general idea of ICE-CI is straightforward: Consider a many-particle state that has at least a sizeable contribution from a given configuration $\mathbf{n}^0$ (this is a set of occupation numbers for the active orbitals that are $n_p^0 = 0, 1\ or\ 2$ (p = any active orbital). By nature of the non-relativisitic Hamiltonian only configurations that differ by at most two orbital occupations from $\mathbf{n}^0$ will interact with it. We can use perturbation theory to select the subset of singles and doubles that interact most strongly with $\mathbf{n}^0$ and then solve the variational problem. We can then analyze the CI vector for configurations that make a dominant contribution to the ground state. Say, we single out the configurations with $C_I^2 > T_{\text{gen}}$. This defines the "generator" set of configurations. The other configurations are called "variational" configurations. They are treated to infinite order by the variational principle, but are not important enough to bring in their single and double excitations. In the next iteration, we perform singles and doubles relative to these general configurations and select according to their interaction with the dominant part of the previous CI vector (truncated to the generators). This procedure can be repeated until no new important configurations are found and the total energy converges (See Fig. 3.24).

Fig. 3.24: Flowchart of the ICE-CI procedure.

The described procedure is very similar to Malrieu's three level CIPSI procedure. One major technical difference, is that ICE is centered around *configurations* and *configuration state functions* rather than *determinants*. A configuration is a set of occupation numbers 0, 1 or 2 that describes how the electrons are distributed among the available spatial orbitals. A *configuration state function* (CSF) is created by coupling the unpaired spins in a given configuration to a

given total spin S. In general there are several, if not many ways to construct a linearly independent set of CSFs. CSFs on the other hand can be expanded in terms of Slater determinants, but there are more Slater determinants to a given configuration than CSFs. For example for a CAS(14,18) calculation one has about 10^9 determinants, but only about $3x10^8$ CSFs and $3x10^7$ configurations. In the configuration based ICE (CFG-ICE) all logic happens at the level of configurations. That is, it is the relationship between two configurations that determines whether and if yes, by which integrals the CSFs or determinants of two given configurations interact. Since the configuration space is so much more compact than the determinant space substantial computational benefit can be realized by organizing the calculation around the concept of a configuration. In general, in CFG-ICE all CSFs that belong to a given configuration are included and all selection quantities are summed over all CSFs of a given configuration before it is decided whether this CSFs is included or not. In the configuration state functions based ICE (CSF-ICE) the logic of generation and selection occurs at the level of individual CSFs and therefore we get rid of the requirement to carry around all the CSFs for a given configuration. This provides substantial gains in the case of molecules containing a large number of transition metal atoms, where each atom contains a high-spin center. In such cases only a few CSFs of a the dominant CFG play a dominant role and other show negligible contribution to the wavefunction. Finally, in some cases the original determinant based CIPSI procedure could be preferred. Such cases can be handled by the determinant based ICE termed DET-ICE. The three variants of ICE therefore cover all the possible types of multi-reference systems that one encounters in quantum chemistry.

It should be noted that although the procedure contains a perturbative element, the final energy is strongly dominated by the variational energy and hence, for all intents and purposes, the ICE-CI procedure is variational (but not rigorously size consistent – size consistency errors are on the same order of magnitude as the error in absolute energy).

3.14.3 A Simple Example Calculation

Let us look at a simple calculation on the water molecule:

```
#
# Check the ICECI implementation
#

! SV

%ice  nel 10           # number of active electrons
      norb 13          # number of active orbitals
      nroots 1         # number of requested roots
      integrals exact  # exact 4-index transformation
                       # can be set to RI to avoid bottlenecks

      icetype   CFGs   # The configuration based ICE-CI
                CSFs   # The CSF based ICE-CI
                DETs   # The determinnat based ICE-CI
                       # please choose one of these three

      Tgen 1e-04       # value for Tgen. Default is 1e-4
      Tvar 1e-11       # value for Tvar. Default is 1e-11 (1e-7*Tgen)

      etol 1e-06       # energy convergence tolerance
end

* int 0 1
O 0 0 0 0.0   0.000 0.000
H 1 0 0 1.0   0.000 0.000
H 1 2 0 1.0 104.060 0.000
*
```

Let us look at the output:

```
----------------------------------------------------------------------------
                ORCA Iterative Configuration Expansion
             - a configuration driven CIPSI type approach -
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------------

(some startup information)

Making an initial 'Aufbau' configuration      ... done
Performing S+D excitations from     1   configs ... done (  0.0 sec) NCFG=581
Performing perturbative selection             ... done (  0.0 sec)
   # of configurations before selection       ... 581
   # of configurations after selection        ... 191
   'rest' energy (probably not very physical) ... -3.736391e-10

                     ******************************
                     *  ICECI MACROITERATION   1  *
                     ******************************

   # of active configurations =    191
   Now calling CI solver (269 CSFs)
(...)
CI SOLUTION :
STATE   0 MULT= 1: E=    -76.0463127108 Eh W=  1.0000 DE= 0.000 eV      0.0 cm**-1
      0.95752  : 2222200000000
   Selecting new configurations         ... done (  0.0 sec)
   # of selected configurations          ...    191
   # of generator configurations         ...     69
   Performing single and double excitations relative to generators ... done (  0.0␣
↪sec)
   # of configurations after S+D         ...  13174
   Selecting from the generated configurations  ... done (  0.1 sec)
   # of configurations after Selection  ...   3827
    Root   0:    -76.046312711       -0.000000063     -76.046312773
(...)
                     ******************************
                     *  ICECI MACROITERATION   3  *
                     ******************************

   # of active configurations =   3866
   Now calling CI solver (9606 CSFs)

CI SOLUTION :
STATE   0 MULT= 1: E=    -76.0539542296 Eh W=  1.0000 DE= 0.000 eV      0.0 cm**-1
      0.95097  : 2222200000000
(...)

             ********* ICECI IS CONVERGED *********
(one final CI)

           *******************************************
           **   ICECI Problem solved in    2.6 sec   **
           *******************************************


FINAL CIPSI ENERGIES
  Final CIPSI Energy Root   0:    -76.053954291 EH
```

From the output the individual steps in the calculation are readily appreciated. The program keeps cycling between variational solution of the CI problem, generation of new configurations and perturbative selection until convergence of the energy is achieved. Normally, this occurs rapidly and rarely requires more than five iterations. The result will be close to the Full CI result.

Let us look at a H_2O/cc-pVDZ calculation in a bit more detail (See Fig. 3.25). The calculation starts out with a single Hartree-Fock configuration. The first iteration of ICE-CI creates the singles and doubles and altogether 544 configurations are selected. These singles and doubles bring in about half of the correlation energy. Already

the second iteration, which leads to 73000 selected CSFs provides a result close to the full CI. At this point up to quadruple excitations from the Hartree-Fock reference have been included. It is well known that such quadruple excitations are important for the correct behavior of the CI procedure (near size consistency will come from the part of the quadruple excitations that are products of doubles). However, only a very small fraction of quadruples will be necessary for achieving the desired accuracy. In the first iteration the procedure is already converged and provides 99.8% of the correlation energy, using 0.5% of the CSFs in the full CI space and at less than 0.2% the calculation time required for solving the full CI problem. Hence, it is clear that near exact results can be obtained while realizing spectacular savings.

H_2O, cc-pVDZ, 1s frozen, 8 electrons, 23 orbitals $T_{Gen}=10^{-4}$, $T_{var}=10^{-11}$

	Energy	N(CFG)	N(CSF)		time(sec)
Hartree-Fock	-76.0260966	1	1		2
	1977 CFG's generated, 544 selected				<0.1
Iter=1	-76.2292278	544	805	S+D	0.3
	98717 CFG's generated, 22471 selected				1.1
Iter=2	-76.2412460	22471	73296	T+Q	17.1
	10677 CFG's generated, 23241 selected				2.2
Iter=3	-76.2412769	23241	75705	P+H	20.1
	10677 CFG's generated, 23241 selected				3.3
Iter=4	-76.2412769	23241	75705	S+O	20.1
					83.5 (1 final CI)
	99.75% of EC	0.57%	0.4%		0.18% of FCI
Full-CI	-76.2418240	4065963	18818646		44529.0

Fig. 3.25: An ICE-CI calculation on the water molecule in the cc-pVDZ basis (1s frozen)

3.14.4 Accuracy

The accuracy of the procedure is controlled by two parameters T_{gen} and T_{var} Since we have found that $T_{var} = 10^{-7}$ T_{gen} always provides converged results, this choice is the default. However, T_{var} can be set manually. It can be reduced considerably in order to speed up the calculations at the expense of some accuracy. Our default values are $T_{gen} = 10^{-4}$ and $T_{var} = 10^{-11}$. This provides results within about 1 mEh of the full CI results (roughly speaking, a bit better than CCSDT for genuine closed-shell systems).

During the development of ICE-CI systematic test calculations have been performed using a reference set of 21 full CI energies on small molecules. The convergence pattern of the mean absolute error is shown in Fig. 3.26. It is evident from the figure that the convergence of ICE-CI towards the FCI result is very smooth and that high accuracy can be obtained. In fact, the default settings lead to an accuracy of <1 mEh deviation to the full-CI result. μEh accuracy can be achieved by further tightening. The achieved accuracy relative to accurate coupled-cluster results shows that the accuracy of even CCSDTQ can be surpassed by ICE-CI. The achievable accuracy is only limited by the value of T_{gen} and much less so by the value of T_{var}. Hence, it is advisable to use a value for T_{var} that is essentially converged and control the accuracy of the procedure by T_{gen}.

Fig. 3.26: Convergence of the ICE-CI procedure towards the full CI results for a test set of 21 full CI energy. Shown is the RMS error relative to the Full CI results. The corresponding errors for various coupled-cluster variants is shown by broken horizontal lines.

3.14.5 Scaling behavior

ICE-CI will break the factorial scaling of the full CI problem and scale polynomially. The actual order of the polynomial scaling is system dependent and accuracy dependent. In order to provide some impression, consider some calculations on linear polyene chains.

Fig. 3.27: Polyene chains used for scaling calculations.

The results are displayed in the Fig. 3.28. It is evident from Fig. 3.28 that ICE-CI breaks the factorial scaling of the full CI problem. In fact, for a thresholds of T_{gen} $=10^{-4}$, 10^{-3} and 10^{-2} the observed scalings are approximately $O(N^8)$, $O(N^7)$ and $O(N^6)$ respectively. These numbers will obviously be very system dependent but should serve as a rough guide. The calculations become quickly much more expensive if T_{gen} is tightened. A rule of thumb is that each order of magnitude tightening of T_{gen} increases the computation time by a factor of 10. The above calculations have been performed on a simple desktop computer and it was already possible to solve a CAS(30,30) problem in less than one day of elapsed time using the default thresholds. Large active spaces will require either loosening of the tresholds or large, more powerful machines.

Fig. 3.28: Scaling behavior of ICE-CI for linear polyene chains (Full π-electron active space) as a functions of system size for different generator thresholds.

3.14.6 Accuracy of the Wavefunction

The accuracy of the many particle wavefunction is not straightforward to check. A reasonable measure, however, is how well it converges towards the exact result for one-electron expectation values. Since every expectation value can be written in terms of natural orbitals of the one-particle density as:

$$\left\langle \hat{O} \right\rangle = \left\langle \Psi \left| \sum\nolimits_o \hat{o}(\mathbf{x}_i) \right| \Psi \right\rangle = \sum_{pq} D_{pq} \left\langle \psi_p | \hat{o} | \psi_q \right\rangle = \sum_p n_p \left\langle \tilde{\psi}_p | \hat{o} | \tilde{\psi}_p \right\rangle$$

where $\hat{o}(\mathbf{x}_i)$ is an arbitrary one-particle operator, D_{pq} is the density matrix of the ICE-CI wavefunction, $\tilde{\psi}_p$ are the natural orbitals of the ICE-CI wavefunction and n_p are their occupations numbers. It is reasonable to take the deviation of the natural orbital occupation numbers as a measure for wavefunction convergence.

For example, we treat the H_2O/cc-pVDZ problem again. From the results in Fig. 3.29 it becomes evident that the ICE-CI wavefunction is fairly accurate. At the default threshold the occupation numbers agree to within 10^{-3} with the full CI reference numbers, which means that expectation values will be of similar accuracy. Interestingly, the largest errors occur in the region of the HOMO-LUMO gap, where apparently all approximate wavefunction approaches tend to depopulate the high lying orbitals too much and put too much electron density in the low lying empty orbitals. From comparison, it is seen, that the CCSD natural occupation numbers for this problem are significantly less accurate. Hence, this is evidence that the ICE-CI wavefunction is properly converging to the right result.

Fig. 3.29: Convergence of the ICE-CI natural orbital occupation numbers. The upper panel is showing the Full CI occupation numbers, the lower panel the deviation of the ICE-CI values from these exact values. For comparison, the CCSD natural orbital occupation numbers are also provided.

3.14.7 Potential Energy Surfaces

You can use ICE-CI to scan entire potential energy surfaces. In general, the non-parallelity error along a potential energy surface is very small. Thus, ICE-CI yields consistent quality throughout the surface.

For example, let us look at the potential energy surface of the N_2 molecule (Fig. 3.30) – a common test case for quantum chemical methods. There are not too many methods that would disscociate the triple bond of N_2 correctly – ICE-CI is one of them. The potential energy surface is entirely smooth and also correctly behaves in the dissociation limit. Near the minimum it is very close to high-level coupled-cluster methods that, however, all fail badly as the bond is stretched.

Fig. 3.30: Potential energy surface of the N_2 molecule in the SV basis. For comparison higher level coupled-cluster results are also shown.

It is interesting to observe the variations of the ICE-CI wavefunction along the dissociation potential energy surface. As an example, we look at the dissociation curve of H_2O where both O-H bonds are simultaneously stretched (Fig. 3.31). It is seen that the ICE-CI method is extremely parallel to the full CI curve at all distances. Hence, the description of the bond remains consistent, even when Hartree-Fock becomes a bad approximation. The agreement is particularly good if MP2 natural orbitals are used in the ICE-CI procedure. With the default value of $T_{gen} = 10^{-4}$ and MP2 natural orbitals the error is consistently below 0.2 mEh. For tighter thresholds, the error is below 0.05 mEh. By contrast, the CCSD(T) method shows relatively large deviations from the full CI results and also behaves very non-parallel as a function of O-H distance.

Fig. 3.31: Non-parallelity error of ICE-CI for the H_2O molecule in the SV basis. Shown is the deviation from the full CI value as a function of O-H distance (both bonds stretched). For comparison, the CCSD(T) curve is also shown

It is instructive to analyze the ICE-CI wavefunction along the dissociation pathway (Fig. 3.32). It becomes apparent that the wavefunctions stays compact along the entire surface, even in the dissociation limit, where the weight of the Hartree-Fock wavefunction drops to less than 25%. Even in this drastic limit, the ICE-CI wavefunction consists of only about 60000 CSFs, which is very similar to the size of the wavefunction at equilibrium geometry. As the wavefunction becomes more multiconfigurational, the number of generator configurations goes slightly up from the equilibrium value of 77 to a maximum of 118 and finally 112 at dissociation. It is also interesting to note that along the entire dissociation pathway no configuration with more than 8 open shells is generated, which means that no more than quadruple excitations are contained in the ICE-CI wavefucntion. The number of iterations required in the ICE-CI procedure also stays constant along the surface at 4 iterations, which impressively shows that a dominant configuration is not necessary for a successful ICE-CI calculation.

Fig. 3.32: Analysis of the ICE-CI wavefunction along the O-H dissociation pathway.

3.14.8 Excited States

ICE-CI can be used to obtain some insight into excited states starting from no knowledge at all. Of course, the best was to start an excited state calculation is to have some idea which configurations are important for the low-lying states of the system. If this is not the case, an automated procedure is used. The program will first generate an "Aufbau" configuration using the orbitals that are provided on input. Starting from this Aufbau configuration, single excitations at the configuration level are performed an the Hamiltonian is diagonalized for the required number of roots. These roots are then analyzed for the leading configurations and the regular ICE-CI procedure is started from those configurations. For example, look at a calculation on the CN radical. In this case, we know the relevant orbitals and leading configurations for the lowest four roots (a doublet Σ ground state, a doublet Π excited state and a doublet Σ excited state) and hence can provide them in the input file as shown below.

```
#
! cc-pVDZ VeryTightSCF

%maxcore 4096

%casscf nel 7
        norb 4
        nroots 4
        mult 2
        end

%ice   nel 9
        norb 26
        nroots 4
        cimode 3
        tvar 1e-11
        tgen 1e-4
```

(continues on next page)

(continued from previous page)

```
        refs { 2 2 2 2 1 }
             { 2 2 2 1 2 }
             { 2 2 1 2 2 }
             { 2 1 2 2 2 }
             end
        end

* xyz 0 2
C 0 0 0
N 0 0 1.07
*
```

The result is shown below. The excitation energies are reasonable but not highly accurate due to the limitations of the basis set (experimentally the doublet Π state is at 1.32 eV and the doublet Σ state at 3.22 eV). There is a very slight symmetry breaking In the doublet Π state that arises from the selection procedure. It should be noted that the state averaged CASSCF excitation energies are 0.25 eV and 3.18 eV.

```
STATE   0 MULT= 2: E=    -92.4542949092 Eh W=  0.2500 DE= 0.000 eV      0.0 cm**-1
      0.67408  : 22221000000000000000000000
STATE   1 MULT= 2: E=    -92.3777338429 Eh W=  0.2500 DE= 2.083 eV  16803.2 cm**-1
      0.65997  : 22212000000000000000000000
      0.12092  : 22122000000000000000000000
      0.11545  : 22221000000000000000000000
STATE   2 MULT= 2: E=    -92.3776916150 Eh W=  0.2500 DE= 2.084 eV  16812.5 cm**-1
      0.69860  : 21222000000000000000000000
      0.16565  : 22122000000000000000000000
STATE   3 MULT= 2: E=    -92.3412510872 Eh W=  0.2500 DE= 3.076 eV  24810.3 cm**-1
      0.51251  : 22122000000000000000000000
      0.16825  : 22212000000000000000000000
      0.11151  : 21222000000000000000000000
```

Below, it is described how to do ICE-CI calculations on excited states if the dominant configurations are not known.

3.14.9 Tips and Tricks

ICE-CI can be used very fruitfully together with, say, MP2 natural orbitals. This usually results in results that are closer to full CI results and at the same lead to more compact wavefunctions (it may be called nICE). The use of MP2 natural orbitals is requested by choosing `UseMP2nat true` inside the %ice block. Alternatively, improved virtual orbitals can be used (requested by `UseIVOs true`). A comparison is shown in Scheme *Comparison of MP2 natural orbitals and improved virtual orbitals for the ICE-CI procedure (H2O molecule, cc-pVDZ basis, equilibrium geometry)*. It is evident that the calculations based on the MP2 natural orbitals show an error relative to full CI that is almost a factor of two smaller than the corresponding result with canonical orbitals while at the same time the wavefunction is more compact by more than 30%. Hence, the use of MP2 natural orbitals appears to be a very good idea in conjunction with the ICE-CI procedure. This also holds when MP2 itself is a bad approximation (for example in the dissociation limit of the H2O molecule as shown above). On the other hand, the IVOs behave very similar to canonical orbitals and hence, seem to offer fewer advantages.

	#(Iterations)	#(CSF)	Energy/Eh	Error/μEh
Canonical orbitals:	4	85220	-76.241 306 7	518
Improved virtual orbitals:	4	85064	-76.241 316 6	508
MP2 natural orbitals:	4	59447 (0.3% of FCI!)	-76.241 434 0 (99.82% of E_{CI}!)	390
Full CI result		18818646	-76.241 824 0	

Fig. 3.33: Comparison of MP2 natural orbitals and improved virtual orbitals for the ICE-CI procedure (H2O molecule, cc-pVDZ basis, equilibrium geometry)

If ICE-CI is used in conjunction with MP2 natural orbitals, there also is the possibility of letting the program automatically choose the active space (this is called auto-ICE). The general idea is simple – we base the active space on the MP2 natural orbitals and their occupation numbers. All orbitals between occupation number say 1.98 down to 0.02 will be included in the active space. A relevant input is shown below.

```
! cc-pVDZ aug-cc-pV6Z/C Auto-ICE
%ice nmin 1.99 nmax 0.01 end

%paras R= 1.0 end

* int 0 1
O 0 0 0 0 0 0
H 1 0 0 { R} 0 0
H 1 2 0 { R} 104 0
*
```

If we scan along the H_2O dissociation surface one can see that despite changing active spaces, the dissociation curves are smooth and remain fairly parallel to the full CI dissociation curve. Depending on the tightness of the thresholds the active space may change from a small 6 electrons in 5 orbitals to a larger 8 electrons in 7 or 8 orbitals upon dissociation. This is the expected behavior as the σ-antibonding orbital becomes more stable along the bond stretching coordinate. Hence, these results are encouraging in as far as in many situations the program will be able to select a sensible active space without extended input from the user.

Fig. 3.34: Automatic active space selection along the H_2O dissociation surface. The reference curve (blue triangles) is the ICE-CI method for the full orbital space with the default parameters.

Another place, where automatic selection comes in conveniently is in the calculation of excited states. If there are no user supplied configurations, what happens is that the program will first choose an Aufbau "reference" configuration and then perform all single excitations relative to this configuration. The program will then diagonalize the Hamiltonian over the this set of configurations to create 0th order approximations for the chosen number of roots of interest and then initiate the ICE-CI procedure starting from the leading configurations of these states. Here is an example for the benzene molecule:

```
! RHF def2-SVPD def2-SVP/C Auto-ICE
%cclib "/user/me/orca/cclib/orcacc"
%ice nroots 5
     nmin 1.98
     nmax 0.02
     integrals ri
     end

* int 0 1
C 0 0 0 0.000000 0.000 0.000
C 1 0 0 1.389437 0.000 0.000
C 2 1 0 1.389437 120.000 0.000
C 3 2 1 1.389437 120.000 0.000
C 4 3 2 1.389437 120.000 0.000
C 5 4 3 1.389437 120.000 0.000
H 1 2 3 1.082921 120.000 180.000
H 2 1 3 1.082921 120.000 180.000
H 3 2 1 1.082921 120.000 180.000
H 4 3 2 1.082921 120.000 180.000
H 5 4 3 1.082921 120.000 180.000
```

(continues on next page)

(continued from previous page)

```
H 6 5 4 1.082921 120.000 180.000
*
```

(The %cclib statement is explained below and is not mandatory here). The Auto-ICE procedure comes up with as many as 24 electrons in 19 orbitals, which already is a fairly heavy calculation. The procedure converges in five iterations and provides indeed the correct states: the gorund state, the $^1B_{2u}$ state at 6.4 eV, the $^1B_{1u}$ state at 8.9 eV and a degenerate $^1E_{1u}$ state at 10.0 eV. These excitation energies are still in error by about 2 eV relative to experiment, which is mainly due to missing dynamic correlation. However, the correct states and their sequence has been found.

The ICE-CI can be used to find the ground state if the actual ground state is not known. To this end, one simply has to turn off the selection steps. This makes the calculations more expensive, but they will converge to the lowest state. In the example below (again, the H_2O molecule) we start from a random quintuply excited configuration – the ICE-CI still finds the ground state after four iterations:

```
%ice
  nel     8
  norb    23
  nroots   1
  tvar 1e-11
  tgen 1e-04
  etol 1e-06

  # selection
  SelStart false
  SelIter  false

  # algorithm details
  useivos   false
  integrals exact
  cimaxdim     5 #Davidson expansion space = MaxDim * NRoots
  cimode       3

  # spatial sym (only supported for ICEType = CFGs)
  irrep 0
  # startup (optional)
  refs { 2 1 0 1 0 2 1 0 1 }
       end
end
```

However, if one wants to converge to an excited state, one should turn on the selection. In the example below (once more the water molecule) one can converge to the second excitated singlet state by judicious choice of the start configuration:

```
%ice
  nel 8
  norb 23
  nroots 1
  tvar 1e-11
  tgen 1e-04
  etol 1e-06

  # selection
  SelStart true
  SelIter true

  # algorithm details
  useivos false
  integrals exact
  cimaxdim 5
  cimode 3
```

(continues on next page)

(continued from previous page)

```
  # spatial sym (only supported for ICEType = CFGs)
  irrep 0
  { #} startup (optional)
  refs
    { 2 2 1 2 0 1 }
  end
end
```

3.14.10 Large-scale approximate CASSCF: ICE-SCF

ICE-CI can be used in place of the default `CIStep` in a *CASSCF* framework. In this way, much *larger CASSCF calculations* than previously possible can be envisioned. In using the ICE-CI in this way, the active orbitals should be chosen as natural orbitals in order to ensure a proper canonicalization. In general, ICE-CI results will not be invariant with respect to the choice of orbitals. However, in practice we have not found this to be problematic. We refer to this as ICE-SCF.

The use is simple: In the `%casscf` block choose `CIStep ICE`. Optionally, starting configurations (CFGs) or configuration state functions (CSFs) can be defined in the `refs` sub-block. All other parameters are accessible through the 'CI' sub-block in `%casscf`. The program also supports *FIC-NEVPT2 calculations with the ICE* as reference wavefunction.[459]

```
%casscf
...
  # mandatory
  CIStep ICE

  # optional starting references for ICE
  # using the refs sub-block
  # refs {2 2 ...} end

  # optional refined settings
  # with keywords as in the %ice block
  ci
     ICEType CFGs   # select ICE expansion type:
                    # 'CFGs' (default), 'DETS' or 'CSFs'
     TGen 1e-4      # controls accuracy (default = 1e-4)
     TVar 1e-11     # default = 1e-7 * TGen
     MaxIter 100    # number of allowed cycles (default = 64)

     # Scratch space dimension (number of singles/double excitations per CSF)
     # If scratch is too small, the program may crash.
     CIBufferLength 10240 # ~82kB (factor 8/10^3 to get the memory in kB):
  end
end
```

The entire remaining input is the one for standard CASSCF calculations. In this way one can do CASSCF calculations with very large active space in reasonable turnaround times. We have not observed convergence problems that are worse than in the standard CASSCF procedure. The results in Fig. 3.35 show that the deviations from regular CASSCF energies are very small. The largest deviation observed for C_2H_4 is on the order of 0.2 mEh, which appears acceptable. Note that the CASSCF tutorial also covers larger examples and excitations energies computed with the ICE-CI as CI solver.

Fig. 3.35: Deviations of ICE-SCF from CASSCF energies for a selection of molecules (standard truncation parameters $T_{\text{gen}} = 10^{-4}$ and $T_{\text{var}} = 10^{-11}$)

Since CASSCF is fully variational, it is possible to optimize geometries with that procedure. It is our experience so far, that the ICE-SCF geometries are virtually indistinguishable from CASSCF geometries (an example is shown in Fig. 3.36).

Fig. 3.36: CASSCF and ICE-SCF optimized geometries for methylene and ozone (cc-pVDZ basis set, default parameters).

3.14.11 Keywords

For completeness, the parameters that can be specified in the input block are summarized below:

```
%ice
        nel             8 # number of active electrons
        norb           23 # number of active orbitals
        nroots          1 # number of roots
        mult            1 # requested multiplicity
        irrep           0 # requested irrep (only supported for ICEType = CFGs)only␣
↪s
        tgen       1e-04 # generator threshold
        tvar       -1e-7 # negative -> 1e-7*tgen
        etol       1e-06 # convergence tolerance

        icetype     CFGs # The configuration based ICE-CI
                    CSFs # The CSF based ICE-CI
                    DETs # The CSF determinant based ICE-CI

        # algorithm details


        useMP2nat  false  # use MP2 natural orbitals
        useIVOs    false  # use improved virtual orbitals
        useQROs    false  # For UHF: use quasi-restricted MOs?
        integrals  exact  # exact or ri transformation
        CIMaxIter     64  # max. number of CI iterations in the Davidson procedure
        CINGuessMat   512 # size of the CI guess matrix in the Davidson procedure
        CIMaxDim      10  # max. size of expansion space in the Davidson procedure
        CIMode         3  # default=accelerated CI, other settings not recommended
        CSFCIwithRI    1  # 1=with RI, 0=without RI, use withour RI when norb >>␣
↪nel (e.g. (30e, 120o))
        CIPSIOrbSweepWindow 1 # Use in case of large size of the S+D list and␣
↪small available memory.
                              # MOs can be divided into chunks (e.g. MOblock = MO/
↪CIPSIOrbSweepWindow)

        # scratch space dimension (number of singles/double excitations per CSF)
        # If scratch is too small, the program may crash.
        CIBufferLength 10240 # ~82kB (factor 8/10³ to get the memory in kB):

        # startup configurations(optional)
        refs { 2 2 2 2 0 }
             { 2 2 2 0 2 }
             end
        end

        # natural orbitals with the ICE ansatz
        NatOrbs 1 # generates the .cipsi.nat GWB file containing the natural␣
↪orbitals
```

3.14.12 A Technical Note: orcacclib

We should finally mention a technical aspect. The CI procedure in ICE-CI is based around the so-called one particle coupling coefficients

$$A_{pq}^{IJ} = \langle I|E_p^q|J\rangle$$

where A_{pq}^{IJ} is a coupling coefficient, I and J are configuration state functions (CSFs) and E_p^q is the spin-free excitation operator that promotes an electron from orbital p to orbital q. The values of these coupling coefficients only depend on the logical relationship between the CSFs I and J but not on the absolute values of I, J, p, q. In fact, they only depend on the number of unpaired electrons in I and the total spin S that both CSFs refer to. Hence, prototype coefficients can be pre-tabulated. This is normally done in a CI run at the beginning of the run. However, in ICE-CI it may have to be repeated several dozen times and for large numbers of open shells (say 14), the process is time and memory consuming.

In order to ease the computational burden, we have provided a small utility program that tabulates the coupling coefficients for a given total spin S (rather the multiplicity $M = 2S + 1$) and maximum number of open shells. This program is called `orcacclib`. It is called like:

```
orca_cclib Mult MaxNOpen
```

or - if you want to speed up the generation of the cclib:

```
mpirun -np 4 /full_path/orca_cclib_mpi Mult MaxNOpen cclib  # using 4 processes
```

It will produce a series of files orcacc.el.mult.nopen (electron density coupling coefficients) and orcacc.sp.mult.nopen (spin-density coupling coefficients) in the current directory. These files are binary files. They can be copied to an arbitrary directory. You instruct the program to read these coefficients (rather than to recalculate them all the time) by setting the path to this directory:

```
# My Job
! def2-SVP Auto-ICE
%cclib "/user/me/orca/cclib/orcacc"
```

The remaining part of the filename will be automatically added by the program. This option can save humongous amounts of time. The coupling coefficient library needs to be made for the desired multiplicities only once. The practical limit will be 14-16 open shells. If you are running the calculation on a cluster using some submit script, you have to ensure that the provided `cclib` path is accessible from the compute node.

3.15 Density Matrix Renormalization Group (DMRG)

The `BLOCK` code in ORCA is only available on the Linux platform!

`BLOCK` is an implementation of the density matrix renormalization group (DMRG) algorithm from the Chan group. [445, 446, 447, 448, 460] The references given should be cited when using this part of the program. `BLOCK` is platform dependent and several crashes have been reported in the past. Please contact the `BLOCK` developers when you are affected.

The DMRG is a variational wavefunction method. It can be viewed as (i) an efficient method for strong correlation in large complete active spaces, (ii) a brute force method to systematically approach FCI for a large number of electrons and orbitals, (iii) a polynomial cost route to exact correlation in pseudo-one-dimensional molecules, such as chains and rings.

Although the algorithm is somewhat complicated compared to many quantum chemistry methods, significant effort has been devoted in `BLOCK` to ensure that it can be run in a simple black-box fashion. In most cases, only a single keyword needs to be specified.

To provide an idea of how the DMRG can be used, here are some examples. The timings will vary depending on your computational setup, but the following are calculations that run in a few hours to a day, on a single 12-core Xeon Westmere cluster node:

- Complete active space (CAS) CI calculations for active spaces with up to 30 electrons in 30 active orbitals, targetting up to 1–10 states, e.g. Jacobsen's catalyst in a 32 electron, 25 orbital active space,
- One-dimensional chain molecules, with "widths" of up to 4 orbitals, and about 100 orbitals in total, e.g. the π-active space of a 4×25 graphene nanoribbon,
- FCI benchmark solutions in molecules with fewer than 20 electrons, and up to 100 orbitals, e.g. C_2 in a cc-pVTZ basis, D_{2h} symmetry (12 electrons in 60 orbitals),
- Accuracies in energy differences or total energies of about 1 kcal/mol.

The following are calculations which are possible with the BLOCK code, but which are challenging, and require large memory (e.g. up to 8 GB per core) and computational time (e.g. from a day to more than a week on up to 6 12-core Xeon Westmere nodes),

- Complete active space (CAS) CI calculations in active spaces with around 40 electrons in 40 active orbitals, targetting a few states, for example, an Fe(II)-porphine (40 electrons in 38 orbitals) with an active space of Fe 3d, 4d and all porphine π and σ donor orbitals, or an Fe 3d, S 3p active space calculation for $[Fe_4S_4(SCH_3)_4]^{2-}$,
- One-dimensional chain molecules, with "widths" of up to 6 orbitals, and about 100 total orbitals,
- Champion FCI benchmark solutions in small molecules, such as butadiene in a cc-pVDZ basis (22 electrons in 82 orbitals),
- Accuracies in energy differences or total energies of about 1 kcal/mol.

If any these calculations interest you, then you might want to try a DMRG calculation with BLOCK!

3.15.1 Technical capabilities

Currently, BLOCK implements the following

- An efficient DMRG algorithm for quantum chemistry Hamiltonians
- Full spin-adaptation (SU(2) symmetry) and Abelian point-group symmetries
- State-averaged excited states

Note that the standalone version of BLOCK may provide more capabilities than are available through the external interface. See the BLOCK website for details [461].

3.15.2 How to cite

We would appreciate if you cite the following papers in publications resulting from the use of BLOCK :

- G. K.-L. Chan and M. Head-Gordon, *J. Chem. Phys.* **116**, 4462 (2002),
- G. K.-L. Chan, *J. Chem. Phys.* **120**, 3172 (2004),
- D. Ghosh, J. Hachmann, T. Yanai, and G. K.-L. Chan, *J. Chem. Phys.*, **128**, 144117 (2008),
- S. Sharma and G. K-.L. Chan, *J. Chem. Phys.* **136**, 124121 (2012).

In addition, useful DMRG references relevant to quantum chemistry can be found in the review below by Chan and Sharma.

- G. K-.L. Chan and S. Sharma, *Ann. Rev. Phys. Chem.* **62**, 465 (2011),

3.15.3 Basic Usage

Within ORCA, the BLOCK program is accessed as part of the CASSCF module. BLOCK can be run in two modes: CASCI mode (no orbital optimization) or CASSCF mode. To enable CASCI mode, set maxiter 1.

```
%casscf
   maxiter 1 # remove if doing CASSCF
   CIStep DMRGCI
   ...
   end
```

For small molecule CASCI it may be possible to correlate all orbitals. In general, similar to a standard CASSCF calculation, it is necessary to select a sensible active space to correlate. (See Section *Orbital optimization* on CASSCF). This is the responsibility of the user.

Once the orbitals to correlate have been chosen, and the wavefunction symmetries and quantum numbers are specified, the accuracy of the DMRG calculation is governed by two parameters: the maximum number of renormalized states M; and, the order and localization of the orbitals.

The most important parameter in the DMRG calculation is M, the number of renormalized states. This defines the maximum size of the wave-function expansion, which is $O(M^2)$ in length in the renormalized basis. As M is increased, the DMRG energy converges to the exact (FCI or CASCI) limit.

The DMRG maps orbitals onto a 1D lattice, thus the best results are achieved if strongly interacting orbitals are placed next to each other. For this reason, the DMRG energy is not generally invariant to orbital rotations within the active space, and orbital rotation and ordering can improve the DMRG energy for a given M. As M is increased, the DMRG energy becomes less and less sensitive to the orbital ordering and localization.

To minimize the number of wavefunction optimization steps, it is often advantageous to perform DMRG calculations at small M, then increase M to the final maximum value. This sequence of optimizations is governed by the *sweep schedule*, which specifies how many optimization steps (sweeps) to perform at each intermediate value of M.

The above may seem to make running a DMRG calculation more complicated than a usual quantum chemistry calculation, however, BLOCK provides a set of default settings which eliminate the need to specify the above parameters by hand. We highly recommend that you first learn to use the BLOCK program *with these default settings*. In the default mode, the orbitals are ordered automatically (Fiedler vector method [462, 463, 464, 465]) and a default sweep schedule is set.

An example of a default CASCI calculation on the C~2~ molecule correlating all electrons in a VTZ basis, is given here:

```
!cc-pvtz pal4
%MaxCore 16000
%casscf
    nel 8
    norb 58
    nroots 1
    mult 1
    maxiter 1
    CIStep DMRGCI
    DMRG
        maxM 5000
    end
end

* xyz 0 1
C 0 0 -0.621265
C 0 0  0.621265
*
```

Once you are familiar with the default mode, we recommend exploring the localization of orbitals. In general, DMRG benefits from the use of localized orbitals, and these should be used unless the high-symmetry of the molecule (e.g., D_{2h} symmetry) provides compensating computational benefits. We recommend using "split-localized" orbitals, which

correspond to localizing the occupied and virtual orbitals separately. An example of a split-localized default DMRG calculation on the porphine molecule, correlating the full π-space (26 electrons in 24 orbitals), in a cc-pVDZ basis is given in Sec. *Example: Porphine \pi-active space calculation*.

For a given `maxM`, it can take a long time to tightly converge DMRG calculations (e.g. to the default 1e-9 tolerance). To decrease computation time, you may wish to loosen the default tight sweep tolerance or control the maximum number of sweep iterations with the commands `sweeptol` and `maxIter`.

Orbital optimization

Orbital optimization (mixing the external/internal space with the active space, not to be confused with orbital rotation and ordering in the active space) in DMRG calculation can be performed by using the `BLOCK` program as the "CIStep" within a CASSCF calculation, as described above. For the moment, spin-densities and related properties are not available for this `CIStep`.

During the optimization iterations it is important that the active orbitals maintain their overlap and ordering with previous iterations. This is done using `actConstrains`. This flag is set by default.

```
%casscf
ActConstrains  1 # maintain shape and ordering of active orbitals
...
end
```

In general, performing a DMRG calculation with orbital optimization is quite expensive. Therefore, it is often best to carry out the orbital optimization using a small value of `maxM` (enabled by the default parameters `maxM=25` and the resulting sweep schedule), and to carry out a final single-point calculation using a larger value of `maxM`.

Advanced options

There may be times when one wants finer control of the DMRG calculation. All keywords are shown in the *complete set of BLOCK options Keywords* below. The `startM` command allows to change the starting number of states in DMRG calculations. It is also possible to specify the entire sweep schedule manually. A sweep schedule example follows:

```
%casscf
   ...
  dmrg

    MaxIter          14
    switch_rst       1e-3
    TwoDot_to_oneDot 12
    NSchedule        3
    sche_iteration   0,     4,    8
    sche_M          50,   100,  500
    sche_sweeptol 1e-4,  1e-6, 1e-9
    sche_noise    1e-8, 1e-11,  0.0

  end
end
```

The commands above are:

- `MaxIter`, corresponds to the maximum number of sweeps done by DMRG;
- `NSchedule`, specifies the total number of schedule parameters we will specify;
- `Sche_iteration`, details the sweep number at which to change the parameters of the calculation. Notice count begins at 0;
- `Sche_M`, is the number of renormalized states at each sweep;
- `Sche_sweeptol`, is the tolerance of the Davidson algorithm;

- `Sche_noise`, is the amount of perturbative noise we add each sweep;
- `Twodot_to_onedot`, specifies the sweep at which the switch is made from a twodot to a onedot algorithm. The recommended choice is to start with twodot algorithm and then switch to onedot algorithm a few sweeps after the maximum M has been reached. To do a calculation entirely with the twodot or the onedot algorithm, replace the `twodot_to_onedot` line with `twodot 1` or `onedot 1`;
- `switch_rst`, defines the switching threshold of orbital gradient below which DMRG turns to onedot algorithm and restarts from previous operators and wavefunction. This is essential to avoid oscillation of energy values in the orbital optimization.

The default DMRG sweep schedule is selected automatically according to the choice of computational mode. By default two different sets of predefined schedules are supported for CASCI and CASSCF computations, respectively.

In CASCI mode, the default schedule corresponds to the following: starting from a given `startM` (where the default is 250 and 8 sweeps), increase to a value of 1000 (8 sweeps) and increment by 1000 every 4 iterations until `maxM` is reached. The algorithm switches from twodot to onedot two sweeps after the `maxM` has been reached.

In CASSCF mode, the orbital optimization requires much fewer renormalized states to converge the wavefunction with respect to orbital rotations. The default schedule therefore starts with `startM` (where the default is 25 and 2 sweeps), and increments by a factor of 2 every 2 sweeps util `maxM` is reached. The algorithm continues the sweep at `maxM` by decreasing the Davison tolerance `sche_sweeptol` and noise level `sche_noise` every 2 cycles by a factor of 10, until `sche_sweeptol` becomes smaller than `sweeptol`.

For better control of the orbital ordering, we also provide a genetic algorithm minimization method of a weighted exchange matrix. The genetic algorithm usually provides a superior orbital ordering to the default ordering, but can itself take some time to run for large numbers of orbitals. The genetic algorithm can be enabled by

```
%casscf
   ...
   DMRG
     auto_ordering GAOPT
   end
end
```

within the `%casscf` input.

Troubleshooting

The two most common problems with DMRG calculations are that (i) convergence with `maxM` is slower than desired, or (ii) the DMRG sweeps get stuck in a local minimum. (i) is governed by the orbital ordering / choice of orbitals. To improve convergence, turn on the genetic algorithm orbital ordering.

If you suspect (ii) is occurring, the simplest thing to do is to increase the starting number of states with the `startM` (e.g. from 500 to 1000 states). Local minima can also sometimes be avoided by increasing the noise in the DMRG schedule, e.g. by a factor of 10. To check that you are stuck in a local minimum, you can carry out a DMRG extrapolation (see extended Manual in the `BLOCK` website).

Note that the present DMRG-SCF establishes the input order of active space orbitals according to their Hartree-Fock occupancy, even if these orbitals are ultimately canonical or split-localized canonical in nature. This is specified by `hf_occ` in which the Hartree-Fock occupancy is derived by default from the one-electron integrals. Other options for obtaining the occupancy are available (see *Keywords*).

Somet times the energy values produced from one SCF cycle to another may oscillate. Such a nonlinear numerical behaviour may occur typically by the last few iterations, most likely caused by the loss of a certain distribution of quantum numbers (eg, particle number, irrep symmetry and spin) in the blocking and decimation procedure due to incomplete many-body basis. On the other hand, the loss of quantum numbers is the main source of energy discontinuities on potential energy curves calculated by DMRG-SCF using a small number of renormalized states.

In the current release of DMRG-SCF implementation, the number of quantum states is locked to avoid these problems. The locking mechanism is turned on when the orbital gradient falls below a certain threshold defined by the keyword `switch_rst` (default: 0.001). The DMRG calculation then starts from previous operators and wavefunction in which a perturbative noise is not added. Locking quantum states and restaring DMRG wavefunction not only

ensures a smooth convergence towards the final energy but also minimizes the number of iterations. Note that the locking procedure introduces an arbitrariness to the final energy, when a very small M is used, since the final digits of energy depend on where the locking begins. It is therefore not recommended to start locking too early in iterations which could trap the orbital solution in a local mimimum. Finally the quality of resulting orbitals can be checked by carrying out a DMRG calculation with sufficient renormalized states. Using the default value of `switch_rst` DMRG-SCF usually results in the orbitals that are good enough to reproduce the CASSCF energy.

3.15.4 Keywords

```
%casscf
 ...
 # enable DMRG as CI solver (mandatory)
 CIStep DMRGCI

 # refined optional settings for DMRG
 dmrg
 startM   25     # CASSCF mode: number of re-normalized states for a singlee root
          250    # CASCI mode: number of re-normalized states for a single root
 maxM     25     # CASSCF mode: number of re-normalized states for a singlee root
          250    # CASCI mode: number of re-normalized states for a single root
 DryRun   false  # just create an input for Block
 SweepTol 1e-9   # energy tolerance for the sweeps
 auto_ordering NOREORDER # auto_ordering is an int. If set to 0
                         # or the alias NOREORDER, the reordering is skipped.
               FIEDLER   # (default) let Block optimize the active orbital ordering
               GAOPT     # let Block optimize the active orbital ordering
                         # using genetic algorithm
 hf_occ   0 # user-defined initial Hartree-Fock occupancy manually
          1 # default: initial Hartree-Fock occupancy based on the values of
              the one-electron integrals
          2 # initial Hartree-Fock occupancy based on the energy ordering
              of canonical orbitals

 TwoDot_to_OneDot  1  # Switch from two-dot expressions to one-dot
 OneDot  0   # Only one-dot expressions. %In CASCI mode only.
 TwoDot  0   # Only two-dot expressions. %In CASCI mode only.
 switch_rst 1e-3   # Specify the threshold of orbital gradient below which DMRG
                     swithches to one-dot expression by reading previous␣
↪wavefunction.
 warmup  1   # wilson warm-up type
         2, 3 or 4   # n=3 is the default option.
                       The full configuration space of the n sites next to the␣
↪system
                       constitutes the environment states in the warm-up.
                       The remaining sites use the Hartree-Fock guess occupation
 nonspinadapted 0   # default: spin-adapted DMRG
                1   # non-spin-adatped DMRG in which the spin-density calculation
                      is available


 # Define a schedule for DMRG
 MaxIter   14   # Specify maximum number of iterations
 NSchedule -1   # default sweep schedule in CASSCF mode
            0   # default sweep schedule in CASCI mode
           >0   # Number of manual sweep schedule parameters
                # All schedule parameters must be set if this flag is set manually!
 sche_iteration 0, 4, 8   # vector with sweep-number to execute changes
                          # (schedule parameter)

 sche_M        50,100,500   # vector with corresponding M values (schedule␣
```

(continues on next page)

(continued from previous page)

```
↪parameter)
 sche_sweeptol 1e-4,1e-6,1e-9  # vector with sweep tolerances (schedule parameter)
 sche_noise    1e-8, 1e-11,0.0 # vector with the noise level (schedule parameter)

 # Define a separate maxM for DMRG-NEVPT2
 nevpt2_maxm 25   # set maximum number of renormalized states
                    for DMRG-NEVPT2 calculation (default: MaxM)

 end
end
```

3.15.5 Example: Porphine π-active space calculation

We provide a step-by-step basis on localizing the π-orbitals of the porphine molecules and running a CASSCF-DMRG calculation on this system. It will be important to obtain an initial set of orbitals, rotate the orbitals which are going to be localized, localize them, and finally run the CASSCF calculation. We will abbreviate the coordinates as [...] after showing the coordinates in the first input file, but please note they always need to be included.

1. First obtain RHF orbitals:

```
# To obtain RHF orbitals
!cc-pVDZ
* xyz 0 1
N      2.10524     -0.00000     0.00000
N     -0.00114      1.95475    -0.00000
N     -2.14882      0.00000    -0.00000
N     -0.00114     -1.95475     0.00000
C      2.85587     -1.13749    -0.00000
C      2.85587      1.13749     0.00000
C      1.02499      2.75869    -0.00000
C     -1.10180      2.78036     0.00000
C     -2.93934      1.13019    -0.00000
C     -2.93934     -1.13019     0.00000
C     -1.10180     -2.78036    -0.00000
C      1.02499     -2.75869     0.00000
C      4.23561     -0.67410    -0.00000
C      4.23561      0.67410     0.00000
C      0.69482      4.18829    -0.00000
C     -0.63686      4.14584    -0.00000
C     -4.25427      0.70589    -0.00000
C     -4.25427     -0.70589     0.00000
C     -0.63686     -4.14584     0.00000
C      0.69482     -4.18829     0.00000
H      5.10469     -1.31153     0.00000
H      5.10469      1.31153    -0.00000
H      1.36066      5.02946     0.00000
H     -1.28917      5.00543     0.00000
H     -5.12454      1.34852     0.00000
H     -5.12454     -1.34852    -0.00000
H     -1.28917     -5.00543    -0.00000
H      1.36066     -5.02946    -0.00000
C      2.46219      2.41307     0.00000
C     -2.39783      2.44193     0.00000
C     -2.39783     -2.44193    -0.00000
C      2.46219     -2.41307    -0.00000
H      3.18114      3.22163    -0.00000
H     -3.13041      3.24594    -0.00000
H     -3.13041     -3.24594     0.00000
H      3.18114     -3.22163     0.00000
H      1.08819      0.00000    -0.00000
```

(continues on next page)

(continued from previous page)

```
H     -1.13385     -0.00000     0.00000
*
```

2. We then swap orbitals with π-character so they are adjacent to each other in the active space. (π orbitals are identified by looking at the MO coefficients). When they are adjacent in the active space, they can be easily localized in the next step.

```
#To rotate the orbitals (so that we can localize them in the next step)
!cc-pvdz moread noiter
%moinp "porphine.gbw"
%scf
    rotate
    # Swap orbitals
    {70, 72}
    {65, 71}
    {61, 70}
    {59, 69}
    {56, 68}
    {88, 84}
    {92, 85}
    {93, 86}
    {96, 87}
    {99, 88}
    {102, 89}
    {103, 90}
    {104, 91}
    end
end
* xyz 0 1
[...]
*
```

3. After rotating the orbitals, we localize the 13 occupied π-orbitals. This is performed using the `orca_loc` code. The input file follows.

```
porphine_rot.gbw
porphine_loc.gbw
0
68
80
120
1e-3
0.9
0.9
1
```

4. After localizing the occuppied orbitals, we localize the 11 virtual π-orbitals using the `orca_loc` code once again. The input file follows.

```
porphine_loc.gbw
porphine_loc2.gbw
0
81
91
120
1e-3
0.9
0.9
1
```

5. After these steps are complete, we run a CASSCF-DMRG calculation. The standard input file is shown below

```
!cc-pVDZ moread pal4
%moinp "porphine_loc2.gbw"
%MaxCore 16000

%casscf nel 26
norb 24
nroots 1
CIStep DMRGCI
end
* xyz 0 1
[...]
*
```

3.16 N-Electron Valence State Perturbation Theory (NEVPT2)

The N-electron valence state perturbation theory (NEVPT) belongs to the family of internally contracted methods and applies to a CASSCF/CAS-CI type reference wave functions. The NEVPT2 methodology developed by Angeli et al exists in two formulations namely the strongly-contracted NEVPT2 (SC-NEVPT2) and the partially contracted NEVPT2 (PC-NEVPT2). [466, 467, 468] Irrespective of the name "partially contracted" coined by Angeli et al, the latter approach employs a fully internally contracted wave function (FIC). Hence, we use the term "FIC-NEVPT2" in place of PC-NEVPT2. NEVPT2 has many desirable properties - among them:

- It is **intruder state free** due to the choice of the Dyall Hamiltonian [469] as the 0th order Hamiltonian.
- The **0th order Hamiltonian is diagonal** in the perturber space. Therefore no linear equation system needs to be solved.
- It is **strictly size consistent**. The total energy of two non-interacting systems is equal to the sum of two isolated systems.
- It is **invariant under unitary transformations** within the active subspaces.
- "**strongly contracted**": Perturber functions only interact via their active part. Different subspaces are orthogonal and hence no time is wasted on orthogonalization issues.
- "**fully internally contracted**": Invariant to rotations of the inactive and virtual subspaces.

Both methods produces energies of similar quality as the CASPT2 approach.[470, 471] The strongly and fully internally contracted NEVPT2 methods are implemented in ORCA, along with a range of approximations that significantly enhance the methodology's scalability and make it highly attractive for large-scale applications. This includes the ability to handle *large basis sets*, *big molecules*, and *extended active spaces*.

In addition to corrections to the correlation energy, ORCA features a range of spectroscopic properties for NEVPT2, including UV, IR, CD, and MCD spectra, as well as EPR parameters. These properties are computed using the "quasi-degenerate perturbation theory" that is described in section *CASSCF Properties*. The NEVPT2 corrections enter as "improved diagonal energies" in this formalism. ORCA also offers the multi-state extension (*QD-NEVPT2* for the strongly contracted NEVPT2 variant.[472, 473] Here, the reference wave function is revised in the presence of dynamical correlation. For systems, where such reference relaxation is important, the computed spectroscopic properties will improve.

NEVPT2 requires a single keyword on top of a working CASSCF input.

```
!SC-NEVPT2    # for the strongly contracted NEVPT2
!FIC-NEVPT2   # for the fully internally contracted NEVPT2
!DLPNO-NEVPT2 # for the DLPNO variant of the FIC-NEVPT2
! ...
%casscf ...
```

Alternatively, the methods are called within the CASSCF block and detailed settings can be adjusted in the `PTSettings` subblock.

We will go through some of the *detailed setting* in the next few subsections. For historical reasons, a few features, such as the quasi-degenerate NEVPT2, are only available for the strongly contracted NEVPT2. As shown elsewhere, the strong contraction is not a good starting point for linear scaling approaches.[420] Thus newer additions such as the DLPNO and the F12 correction rely on the FIC variant. [474, 475, 476] Note that ORCA by default employs the frozen core approximation, which can be disabled with the simple keyword `!NoFrozenCore`. A complete description of the frozecore settings can be found in section *Frozen Core Options*.

3.16.1 A simple Example: N_2 Ground State

Let us consider the ground state of the nitrogen molecule as a simple example. After defining the computational details of our CASSCF calculation, we insert "`!SC-NEVPT2`" as simple input or specify "`PTMethod SC_NEVPT2`" in the `%casscf` block. Please note the difference in the two keywords' spelling: Simple input uses hyphen, block input uses underscore for technical reasons.

```
!def2-svp nofrozencore PAtom
%casscf nel  6
        norb 6
        mult 1
        PTMethod SC_NEVPT2  # SC_NEVPT2   for strongly contracted NEVPT2
                            # FIC_NEVPT2  for the fully internally contracted␣
↪NEVPT2
                            # DLPNO_NEVPT2 for the FIC-NEVPT2 with DLPNO
                            # DLPNO requires: trafostep RI and an aux basis
end

* xyz 0 1
N        0.0        0.0        0.0
N        0.0        0.0        1.09768
*
```

For better control over the program flow, it is recommended to split the calculation into two distinct parts. First, converge the CASSCF wave function and inspect the resulting orbitals. Then, in a second step, read the converged orbitals and run the actual NEVPT2 calculation.

```
------------------------------------------------------------
                        ORCA-CASSCF
------------------------------------------------------------

...
PT2-SETTINGS:
A PT2 calculation will be performed on top of the CASSCF wave function (PT2 = SC-
↪NEVPT2)
...
------------------------------------------------------------
                         < NEVPT2  >
------------------------------------------------------------
...
 ================================================================
                       NEVPT2 Results
================================================================
   *********************
    MULT 1, ROOT 0
   *********************

  Class V0_ijab :        dE = -0.017748
  Class Vm1_iab :        dE = -0.023171
  Class Vm2_ab  :        dE = -0.042194
  Class V1_ija  :        dE = -0.006806
  Class V2_ij   :        dE = -0.005056
  Class V0_ia   :        dE = -0.054000
```

(continues on next page)

(continued from previous page)

```
  Class Vm1_a   :        dE = -0.007091
  Class V1_i    :        dE = -0.001963

-------------------------------------------------------------
         Total Energy Correction : dE = -0.15802909
-------------------------------------------------------------
         Zero Order Energy       : E0 = -108.98888640
-------------------------------------------------------------
         Total Energy (E0+dE)    : E  = -109.14691549
-------------------------------------------------------------
```

Introducing dynamic correlation with the SC-NEVPT2 approach lowers the energy by 150 mEh. ORCA also prints the contribution of each "excitation class V" to the first order wave function. We note that in the case of a single reference wave function corresponding to a CAS(0,0), the `V0_ijab` excitation class produces the exact MP2 correlation energy.

In section *Example: Breaking Chemical Bonds* the dissociation of the N_2 molecule has been investigated with the CASSCF method. Inserting `PTMethod SC_NEVPT2` into the `%casscf` block we obtain the NEVPT2 correction as additional information.

```
! def2-svp nofrozencore
%casscf nel  6
        norb 6
        mult 1
        PTMethod SC_NEVPT2
end

# scanning from the outside to the inside
%paras
        R = 2.5,0.7, 30
end

*xyz 0 1
N 0.0 0.0 0.0
N 0.0 0.0 {R}
*
```

Fig. 3.37: Potential Energy Surface of the N_2 molecule from CASSCF(6,6) and NEVPT2 calculations (def2-SVP).

All of the options available in CASSCF can in principle be applied to NEVPT2. Since NEVPT2 is implemented as a submodule of CASSCF, it will inherit all settings from CASSCF (`!tightscf`, `!UseSym`, `!RIJCOSX`,...).

3.16.2 RI, RIJK and RIJCOSX Approximation

Setting the RI approximation on the *CASSCF* level, will set the RI options for NEVPT2 respectively. The three index integrals are computed and partially stored on disk. Three index integral with two internal labels are kept in main memory. The two-electron integrals are assembled on the fly. The auxiliary basis must be large enough to fit the integrals appearing in the CASSCF orbital gradient/Hessian and the NEVPT2 part. The auxiliary basis set of the type /J does not suffice here.

```
%casscf
  ...
  TrafoStep RI    # enable RI approximation in CASSCF and NEVPT2
  PTMethod  SC_NEVPT2 # or the NEVPT2 approach or your choice
end
```

Additional speedups can be achieved by approximating the Fock operator using the `!RIJCOSX` or `!RIJK` techniques. When using `RIJCOSX`, an additional auxiliary basis must be specified for the `AuxJ` auxiliary basis slot. For more information on basis set slots, see section *Orbital Basis Sets*.

```
#RIJCOSX one-liner
! def2-svp def2/J RIJCOSX def2-svp/C

# Commented out: alternative definition via %basis block
#%basis
```

(continues on next page)

(continued from previous page)

```
#auxJ "def2/J"
#auxC "def2-svp/C"
#end
```

Whereas the RIJK requires a single auxiliary basis set (AuxJK slot), that is large enough to fit integrals in the Fock-matrix construction, orbital gradient/Hessian and the correlation part. In contrast to COSX, the calculation can also be carried out in conv mode (storing the AO integrals on disk).

```
#RIJK one-liner: !conv is recommended.
! def2-svp def2/JK RIJK

# Commented out: Alternative definition via %basis block
#%basis
#auxJK "def2/JK"
#end
```

The described methodology allows the computation of systems with up to 2000 basis functions. Even larger molecules are accessible in the framework of DLPNO-NEVPT2 described in the next subsection. Several examples can be found in the CASSCF tutorial.

3.16.3 Beyond the RI approximation: DLPNO-NEVPT2

For systems with more than 80 atoms, we recommend the DLPNO-NEVPT2 method, which combines the DLPNO ansatz with the FIC-NEVPT2 approach.[474] This method is particularly effective, as it recovers 99.9% of the FIC-NEVPT2 correlation energies, even for large systems. The input structure for DLPNO-NEVPT2 is similar to that of its parent FIC-NEVPT2 method. Below, we provide an example input for the Fe(II)-complex depicted in Fig. 3.38, where the active space consists of the metal-3d orbitals. The example takes about 9 hour (including 3 hour for one CASSCF iteration) using 8 cores (2.60GHz Intel E5-2670 CPU) for the calculation to finish. A detailed description of the DLPNO-NEVPT2 methodology can be found in our article.[474]. A number of DLPNO specific parameters can be set in the PTSettings sub-block. The respective keywords are otherwise identical with the *DLPNO Coupled Cluster ansatz described elsewhere in the manual*. The most important parameters are TCutPNO, TCutMKNand TCutDO. However, it is recommended to use the simple keywords !TightPNO, !NormalPNO or !LoosePNO to adjust the accuracy. In contrast to the canonical FIC-NEVPT2 method, the DLPNO amplitude equations require the solution of a set of linear equations. The latter are solved iteratively via a DIIS solver, which comes with its own set of control parameters.

```
# DLPNO-NEVPT2 calculation for quintet state of FeC72N2H100
!PAL8 def2-TZVP def2/JK
!moread noiter
%moinp "FeC72N2H100.gbw-CASSCF"
%MaxCore 8000
%casscf
  nel 6
  norb 5
  mult 5

  # mandatory for DLPNO-NEVPT2:
  PTMethod DLPNO_NEVPT2  # will automatically set `TrafoStep RI`


  # optional settings reapting the defaults
  PTSettings
    # DLPNO specific settings (see DLPNO-NEVPT2 article):
    TCutPNO      1e-8 # cutoff for PNO occupation numbers
    TCutPNO_Core 1e-2 # cutoff for PNO occupation numbers for core orbitals
    TCutMKN      1e-3 # cutoff for Mulliken populations in the PNO integral␣
↪transformation
    TCutDO       1e-2 # cutoff for the sparse map construction of domain
```

(continues on next page)

(continued from previous page)

```
    TCutDOij     1e-5 # cutoff for the differential overlap matrix of ij pairs
                      # in the prescreening step

    # DLPNO DIIS solver settings:
    MaxIter    30    # maximum number of iterations
    MaxDIIS    7     # DIIS max. dimension
    DIISDamp1  0.5   # damping of the residual before DIIS is switched ON
    DIISDamp2  0     # damping of the residual after DIIS is switched ON
    LevelShift 0.2   # LevelShift during update
    LMP2TolE   1e-7  # energy convergence criteria
    LMP2TolR   5e-7  # residual convergence criteria
    LMP2Damp   1.0   # damping for the LMP2 amplitude update

  end
end
*xyz 0 5 FeC72N2H100.xyz
```

Just like RI-NEVPT2, the calculations requires an auxiliary basis. The aux-basis should be of /C or /JK type (more accurate). Aside from the paper of Guo et al,[474] a concise report of the accuracy can be found in the CASSCF tutorial, where we compute exchange coupling parameters. Note that in the snippet above, we have commented out the default setting in the `PTSettings` sub-block.

As mentioned earlier, the CASSCF step can be accelerated with the RIJK or RIJCOSX approximation. Both options are equally valid for the DLPNO-NEVPT2. The RIJK variant typically produces more accurate results than RIJCOSX. The input file is almost the same as before, except for the keyword line:

```
# The combination of RIJK with DLPNO-NEVPT2
!PAL8 def2-TZVP def2/JK conv RIJK
```

Fig. 3.38: Structure of the $FeC_{72}N_2H_{100}$

3.16.4 Explicit Correlation: NEVPT2-F12 and DLPNO-NEVPT2-F12

Like in the single-reference MP2 theory, the NEVPT2 correlation energy converges slowly with the basis set. Aside from basis set extrapolation, the R12/F12 method are popular methods to reach the basis set limit. For comparison of F12 and extrapolation techniques, we refer to the study of Liakos et al.[388] ORCA features an F12 correction for the FIC-NEVPT2 wave function using the RI approximation.[475] The RI approximation is mandatory as the involved integrals are expensive. In complete analogy to the single reference MP2-F12, the input requires an *F12 basis*, an *F12-cabs basis* and a sufficiently large RI basis (/JK or /C). It should be noted, that the CASSCF wave function itself is also affected by basis set incompleteness. When `F12` is invoked, a perturbative correction for CASSCF reference, denoted as CABS singles correction, is added.[477]

```
# aug-cc-pvdz/C used as RI basis
 ! cc-pvdz-F12 aug-cc-pvdz/C cc-pvdz-f12-cabs
 %casscf nel 8
         norb 6
         mult 3,1

         # mandatory:
         TrafoStep RI           # RI approximation must be on for F12
         PTMethod FIC_NEVPT2    # FIC-NEVPT2
         PTSettings
           F12 true             # request the F12 correction and CABS singles␣
↪correction.
         end
 end
 *xyz 0 3
  O      0.0 0.0 0.0
```

(continues on next page)

(continued from previous page)

```
 O      0.0 0.0 1.207
 *
```

A linear scaling version of NEVPT2-F12, the DLPNO-NEVPT2-F12, allows to tackle systems with several thousand of basis functions.[475, 476] With the exception of the `DLPNO_NEVPT2` keyword, the input structure is otherwise identical to NEVPT2-F12 method.

```
# aug-cc-pvdz/C used as RI basis
! cc-pvdz-F12 aug-cc-pvdz/C cc-pvdz-f12-cabs
%casscf nel 8
        norb 6
        mult 3,1

        # mandatory:
        TrafoStep RI            # RI approximation must be on for F12
        PTMethod DLPNO_NEVPT2   # DLPNO variant
        PTSettings
          F12 true              # request the F12 correction and CABS singles␣
↪correction.
        end
end
*xyz 0 3
 O      0.0 0.0 0.0
 O      0.0 0.0 1.207
*
```

Note that the DLPNO-NEVPT2-F12 algorithm is unitary invariant with respect to subspace rotation of inactive and active orbitals. By tightening the DLPNO truncation thresholds (`TCutPNO`,`TCutDO`, `TCutCMO` and `TCutDOij`), the canonical NEVPT2-F12 can be reproduced, even with localized internal and active molecular orbitals.

3.16.5 Handling of Reduced Density Matrices

NEVPT2 calculations involve the computation of higher order reduced density matrices (RDMs), which can quickly become a major bottleneck for active spaces of CAS(10,10) and larger. Using Dyall's rank-reduction scheme,[478] the standard implementation involves at most the fourth order reduced density matrix (4-RDM).[468] In contrast, omitting the rank-reduction scheme, the full rank NEVPT2 (FR-NEVPT2) requires the 5-RDM and its contractions.[427] The latter should be used in the context of *approximate CAS-CI reference wave functions such as the ICE*.

ORCA uses an efficient reformulation of the NEVPT2 method, that avoids the explicit 4-RDM and 5-RDM construction.[479] The basic idea is similar to a recent development reported by Sokolov and coworkers.[480] The algorithm requires a memory-intense intermediates, that scales with the number of configuration state functions (CSFs) in the CAS-CI. With the option `D4Step fly`, the memory demands are reduced at the expense of computation time omitting the aforementioned intermediate.

```
%casscf
  ...
  PTMethod     FIC_NEVPT2  # or SC_NEVPT2

  # detailed settings (optional)
  PTSettings
  D4Step efficient # efficient: calling the default RDM handling
                   # fly      : slower but more economic code
                   #
                   # Cu4 : cumulant approximation of the 4-RDM
                   # Cu34: cumulant approximation of the 3-RDM and 4-RDM
  D4TPre 1e-12     # default PS approximation 4-RDM
  D3TPre 1e-14     # default PS approximation 3-RDM
```

(continues on next page)

(continued from previous page)

```
  imaginary 0.0    # imaginary shift (only for FIC-NEVPT2)
end
```

PS Approximation (default)

To accelerate the evaluation of the higher-order RDMs, ORCA uses a prescreening approximation (PS) for the reference wave function.[422] Only configurations with a weight larger than a specified parameter `D4TPre` are taken into account. The same reduction is available for the third order density matrix using the keyword `D3TPre`. Both parameters can be adjusted within the `PTSettings` sub-block of the CASSCF module. The exact NEVPT2 energy is recovered with the parameters set to zero. The approximation is available for all variants of NEVPT2 (SC, FIC and DPLNO-FIC). The default thresholds, `D4TPre 1e-12` and `D3TPre=1e-14`, are chosen very conservative as lowering might lead to false intruder states.[422, 427]

These approximations naturally affect the "configuration RI" as well. In this context, it should be noted that a configuration corresponds to a set of CSFs with identical orbital occupation. For each state the dimension of the CI and and RI space is printed.

```
  D3 Build          ... CI space truncated: 141 -> 82  CFGs
                    ... RI space truncated: 141 -> 141 CFGs
  D4 Build          ... CI space truncated: 141 -> 82  CFGs
                    ... RI space truncated: 141 -> 141 CFGs
```

Cumulant Approximation (optional)

Large computational savings can be achieved with the cumulant expansion, which have been recently re-evaluated.[422] The results should be treated with care as false intruder states can emerge.[481] In these cases, the imaginary level shift is the only mitigation tool.[482] Note that the imaginary shift is only implemented for FIC-NEVPT2.

```
PTSettings
  D4Step Cu4     # Cu4  : approximate the 4-RDM
                 # Cu34 : approximate the 3-RDM and 4-RDM
  imaginary 0.0 # imaginary shift (only for FIC-NEVPT2)
end
```

3.16.6 False Intruder States and Imaginary Shift

False intruder states are introduced by crude approximations of the reduced density matrices or the reference wave function.[422, 427] The occurance is highly system-specific. Indications are unreasonable correlation energy contributions from the 1-hole excitation class (denoted as "V_i" or "ITUV") or the 1-particle excitation class (denoted as "V_a" or "TUVA"), e.g., positive or large correlation energies compared to the 2-hole-2-particle excitation class (denoted as "V_ijab" or "IJAB"). For transparency, ORCA prints warnings when the Koopmans energies or the energy denominators are negative.

```
...
WARNING: Denominator is negative for DEpsilon ITUV i=2 mu=1 : -344.12813836782095
↪< 0                                                                            ␣
↪                                    |
WARNING: Denominator is negative for DEpsilon ITUV i=2 mu=2 : -228.47393616112132
↪< 0                                                                            ␣
↪                                    |
WARNING: Denominator is negative for DEpsilon ITUV i=2 mu=3 : -218.57612632153200
↪< 0                                                                            ␣
↪                                    |
WARNING: Denominator is negative for DEpsilon ITUV i=2 mu=4 : -127.85082085035380
```

(continues on next page)

(continued from previous page)

```
↪< 0                                                                          ␣
↪                                   |
Skipped =       126 of          432 LowestEV=-3.843455e+02 ... done in        0.1␣
↪sec                                                                          ␣
↪                                   |
 Reached max. number of intruder state warning (5). Thus, warnings were suppressed␣
↪for this class to limit the output!                                          ␣
↪                                   |
                                                                              ␣
↪                                                                             ␣
↪                                   |
                                                                              ␣
↪                                                                             ␣
↪                                   |
WARNING: Koopmans matrices have at least one negative eigenvalue (-1.519147e+06)  ␣
↪                                                                             ␣
↪                                   |
         Please check your results carefully.
```

To avoid flooding the output, the number of warnings are limited by `MaxWarnings` (default=5). When faced with these warnings, it is advised to inspect the approximations of the reduced density matrices. For example, in the case of the PS approximation, tightening the thresholds (`D4TPre=1e-14`) should immediately alleviate the issue. The cumulant approximation in particular is prone to false intruder states and should thus be avoided.

As a last resort, an imaginary shift can be added to mitigate intruder states.[482] Note that imaginary shifts (default=0.0) are restricted to the canonical FIC-NEVPT2 implementation.

```
PTSettings
  imaginary 0.2 # imaginary shift (only for FIC-NEVPT2) . default = 0.0
  MaxWarnings 5 # Max number of denominator warnings before going silent.
end
```

3.16.7 Large Active Spaces: ICE-NEVPT2

Standard CAS-CI solver can perform routine calculation up to a CAS(14,14) active space. Larger active spaces are accessible with the *ICE ansatz*.[443] With a recent extension of the FIC-NEVPT2 methodology, that also allows an ICE reference wave functions, calculation with active spaces up to CAS(34,34) are possible.[459] As shown in the paper by Guo et al, for an approximate CAS-CI solution, the canonical implementation, that uses Dyall's rank reduction scheme, cannot be applied without introducing false intruder states. Hence, for an ICE reference wave function, ORCA defaults to the full-rank NEVPT2 formulation (FR-NEVPT2), where additional 5-RDM contributions are accounted for. The latter are enabled with `CASCI_Approx true`. While the default ICE parameter `TVar` is designed for the ICE-CASSCF-Iterations, it is less ideal for the NEVPT2 approach. Here, the computation of the higher-order RMDs is a major bottleneck and not feasible with the same threshold. As demonstrated by Guo et al, a parameter of `TVar 1e-7` (=TGen x 10^{-3}) already leads to reasonable results that balance accuracy and efficiency. Following the reciple from the paper, it is recommended to split the calculation into two parts. First converge the ICE-CASSCF solution with the default parameters and inspect the resulting orbitals. Then, in a second step, run the FIC-NEVPT2 with a larger `TVar` reading the converged orbitals from the first step. For these type of calculation, we recommend to use the `D4Step fly` algorithm as it avoids memory-intense intermedidates processing the RDMs.

```
# Starting with converged CASSCF orbitals
!MORead NoIter
%moinp "converged.ice-casscf.gbw"
%casscf
  ...
  # mandatory keywords for ICE-NEVPT2
  CIStep ICE
  PTMethod FIC_NEVPT2
```

(continues on next page)

(continued from previous page)

```
  CI
    TVar 1e-7 # ICE parameter TVar for ICE-NEVPT2 runs.
              # TVar ~ TGen x 1e-3 - this is larger than the default.
  end


  # optional settings (repeating the defaults)
  PTSettings
    D4Step fly        # memory saving algorithm for larger active spaces
    CASCI_Approx true # setting FR-NEVPT2
    D4TPre 0.0        # recommendation: disabling PS approximation. No benefit␣
↪from screening here.
    D3TPre 1e-14      # recommendation: default PS approximation for 3-RDM.
  end
```

3.16.8 Large Active Spaces: DMRG-NEVPT2

An alternative for larger active spaces is offered by the DMRG ansatz. It is possible to run DMRG-NEVPT2 calculations for the FIC-NEVPT2 and SC-NEVPT2 ansatz using the methodology developed by the Chan group.[483] Here, the reduced density matrices are provided by the BLOCK program. It should be noted that the latter is platform dependent and several crashes have been reported in the past. For troubleshooting, please contact the BLOCK developers directly. Aside from the usual *DMRG input*, the program requires an additional parameter (`nevpt2_MaxM`) in the DMRG block.

```
%casscf
  cistep DMRGCI
  %dmrg
    ...
    nevpt2_MaxM 2000 # (see Note below)
  end
  PTMethod SC_NEVPT2 # or FIC_NEVPT2
end
```

3.16.9 Selecting or Specific States for NEVPT2

ORCA by default computes all states defined in the CASSCF block input with the NEVPT2 approach. There are cases, where this is not desired and the user wants to skip some of these states. The input mask of `SelectedRoots[block]` allows to select only few states for the computation, where "block" refers to numbering of the multiplicity blocks. The enumeration of blocks and roots starts with 0.

```
!NEVPT2 ...
%casscf
  ...
  MULT   3,1 # multiplicity block
  NRoots 2,3 # number of roots for the MULT blocks

  # detailed settings (optional) for the PT2 approaches
  PTSettings
    # option to skip the PT2 correction for all states except for the ones
    # specified for a selected multiplicity block and root
    # selectedRoots[block]  = root number counting from 0.
    selectedRoots[1]=0,1 # compute MULT=1 root=0,1 and skip all others.

  end
end
```

3.16.10 Unrelaxed Densities and Natural Orbitals

For the FIC-NEVPT2 ansatz, it is possible to request state-specific unrelaxed densities

$$\gamma(p,q) = \langle \Psi_I | E_q^p | \Psi_I \rangle,$$

where Ψ_I refers to NEVPT2 wave function of the I'th state and consists of the zeroth- and first-order wave function

$$\Psi_I = |0\rangle + |1\rangle.$$

The code is implemented using the ORCA AGE tool-chain.[362] The exact unrelaxed density requires higher-order reduced density matrices (RDMs) of the the CAS-CI reference wave function. The latter can be avoided, when the unrelaxed density is approximated at first order using the keyword `FirstOrder`. In that case, the expensive second-order contribution $\langle 1|E_q^p)|1\rangle$ is omitted. A cost-effective alternative, that keeps the second-order term, is the cumulant approximation `Cu4` and `Cu34`. Here, the 4-RDM and optionally the 3-RDM themselves are approximated to accelerate their computation.

The unrelaxed density can be used to generate natural orbitals with the keyword `NatOrbs true`. For each state, ORCA stores the resulting orbitals in a separate GBW file with the suffix ".mult.A.root.B.pt2.nat", where "A" flags its multiplicity and "B" its root count. In case of `!UseSymmetry`, the suffix will also include the irreducible representation of the state. NEVPT2 natural orbital are mandatory for various analysis tools, for instance the MultiWFN program. The orbitals can also be used for natural orbital iterations (`!MORead NoIter`) or to systematically extend the active space.[484]

```
%casscf
  ...
  PTMethod     FIC_NEVPT2

  # detailed settings (optional)
  PTSettings
    # densities are disabled by default
    Density Unrelaxed  # unrelaxed density <0+1|E(p,q)|0+1>
            Cu4        # cumulant 4-rdm approximated unrel. density
            Cu34       # cumulant 3/4-rdm approximated unrel. density
            FirstOrder # approximate unrel. density <0|E(p,q)|1>

    NatOrbs True       # produces the natural orbitals (default=False)
  end
end
```

Note that ORCA repeats the population analysis for each state. With the added keyword `!KeepDens` the NEVPT2 density is stored in the density container (**.densities** file on disk) with the prefix "Tdens-CASNEV". The latter can be used to create density plots interactively (see Section *orca_plot*).

Below is a snapshot of a typical output:

```
Unrelaxed Density     ...
Incorizing ADC        ... done in      0.6 sec
Norm <Psi|Psi>        ... done in      0.1 sec (NORM=     1.064186836)
RDM1 <Psi|E|Psi>      ... done in      0.7 sec
Reference Weight      ...     0.939684618
Trace RDM1            ...    20.000000000 (prior correction)

*** Repeating the population analysis with unrelaxed density.
Orbital energies/occupations assumed diagonal. ***
(Note: Temporarily storing unrelaxed densities as cas.scfp)
-------------------------------------------------------------------------------
ORCA POPULATION ANALYSIS
-------------------------------------------------------------------------------


...
Natural Orbital Occupation Numbers:
```

(continues on next page)

(continued from previous page)

```
...
N[  4] =   1.98812992
N[  5] =   1.98308480
N[  6] =   1.93858508
N[  7] =   1.49303660
N[  8] =   1.49303660
N[  9] =   1.48519842
N[ 10] =   1.48519842
N[ 11] =   0.05922342
N[ 12] =   0.00921465
N[ 13] =   0.00921465
N[ 14] =   0.00794869
N[ 15] =   0.00620254
...
 ================================================================
 NEVPT2 Results
 ================================================================
 ...
```

3.16.11 State-averaged SC-NEVPT2

In the definition of the Dyall Hamiltonian,[469] the CASSCF orbitals are chosen to diagonalize the Fock operator (pseudo-canonicalized). Therefore, using a state-averaged CASSCF wave function, the NEVPT2 procedure involves the construction and diagonalization of the "state-specific" Fock operators and is thus resulting in a unique set of orbitals for each state. This becomes quickly inefficient for large numbers of states or larger molecules since each orbital set implies an integral transformation. This is the **default** setting for NEVPT2 and is printed in the output

```
NEVPT2-SETTINGS:
Orbitals      ... canonical for each state
```

For the **SC-NEVPT2**, other orbital options can be set using the keyword `canonstep`.

```
%casscf
  ...
  # detailed settings (optional)
  PTSettings
  # SC-NEVPT2 specific options
  canonstep      0         # state-averaged orbitals and specific orbital energies
                 canonical # canonical for each state (default)
                 averaged  # state-averaged orbitals and orbital energies
                 relaxed   # 1-step orbital relaxation
                           # and canonical for each state (partially relaxed)
  end
end
```

The final orbitals of the state-averaged CASSCF diagonalize the state-averaged Fock operator. Large computational savings can be made if these orbitals are employed for all of the states. `canonstep 0` chooses orbital energies as diagonal elements of the state-specific Fock operators. In release versions ORCA 3.0 and older, this has been the default setting. These options work best if the averaged states are similar in nature. For SC-NEVPT2, we have implemented two more `canonsteps`, which trade accuracy for speed and *vice versa*. `canonstep averaged` is more approximate and employs orbital energies from the state-averaged calculation. Thus there is no contribution to excitation energies from the 2-hole-2-particle perturber class (denoted as "V_ijab" or "IJAB") at this level of approximation.

If the states under consideration are substantially different, these approximations will be of poor quality and should be turned off. Better results can be achieved, if the state-averaged orbitals are partially relaxed for each state before the actual SC-NEVPT2 calculation. [485] Often it is not possible to optimize the orbitals of each excited state separately. Thus, `canonstep relaxed` will try a single steepest descent step for each state before running the actual SC-NEVPT2 calculation with canonicalized orbitals. Optionally, instead of a steepest descent using an approximate

diagonal Hessian, a single Newton-Raphson step can be made.

```
%casscf
  ...
  PTMethod SC_NEVPT2
  # detailed settings
  PTSettings
  canonstep relaxed # orbital relaxation for each state
  gstep SOSCF true  # steepest descent step
  NR    false       # Newton-Raphson   step
  end
end
```

Despite a converged state-averaged calculation, the gradient for the individual states can be surprisingly large. As a consequence, the orbital relaxation may fail, when both methods are outside their convergence radius. ORCA will retry the relaxation with an increased damping. If the orbital update still fails, the program will stick with the initial orbitals. Setting an overall damping manually may help the relaxation procedure.

```
%casscf
  ...
  PTMethod SC_NEVPT2
  # detailed settings
  PTSettings
    gscaling 0.5 # damp gradient with a pre-factor
  end
end
```

3.16.12 Quasi-Degenerate SC-NEVPT2

The NEVPT2 approach presented in the previous subsections follows the recipe of "diagonalize and perturb". The 0th order wave function is determined by the diagonalization of the CAS-CI matrix. The space spanned by the CAS-CI vectors is often referred to as "model space". The subsequent perturbation theory is constructed based on the assumption that the states under consideration are well described within the model space. Consequently, the first order correction to the wave function $\Psi_I^{(1)}$ does not affect the composition of the reference state $|I\rangle$. Corrections to the wave function and energy arise from the interaction of the reference state with the functions $|k\rangle$ of the contributing first order interacting space

$$\Psi_I^{(1)} = \sum_K C_K \left|K\right\rangle$$

$$E_I^{(2)} = \sum_K \frac{\left\langle I \left|H\right| K\right\rangle \left\langle K \left|H\right| I\right\rangle}{E_I^{(0)} - E_K}$$

This is problematic, when the interaction/mixing of states are falsely described at the CASSCF level. A typical example is the dissociation of lithium fluoride.

```
!def2-tzvp nevpt2 nofrozencore

%casscf
  nel 2
  norb 2 #Li(2s), F(2pz)
  mult 1
  nroots 2
end

# unrelaxed scan r=3 to r=7
%paras
r = 3,7,200
end
```

(continues on next page)

(continued from previous page)

```
*xyz 0 1
Li 0 0 0
F 0 0 {r}
*
```

Here, the ground and first excited state of Σ^+ should not cross. However, at the NEVPT2 level, an erratic double crossing is observed instead.

Fig. 3.39: SC-NEVPT2 and QD-SC-NEVPT2 Li-F dissociation curves of the ground and first excited states for a CAS(2,2) reference

A re-organizing of the reference states can be introduced in the framework of quasi-degenerate perturbation theory. In practice, an effective Hamiltonian is constructed allowing "off-diagonal" corrections to the second-order energy

$$H_{IJ} = \delta_{IJ} E_I^{(0)} + \sum_K \frac{\langle I|H|K\rangle \langle K|H|J\rangle}{E_I^{(0)} - E_K}$$

Diagonalization of this eff. Hamiltonian yields improved energies and a rotation matrix (right eigenvectors) that introduces the desired re-mixing of the reference states. The quasi-degenerate extension to SC-NEVPT2 [486] can be switched on with the keyword `QDType`.

```
%casscf
  ...
  PTMethod SC_NEVPT2      # Must be SC_NEVPT2
  PTSettings
    PrintLevel 1          # default printing

    # mandatory
    QDType QD_VanVleck # QD_VanVleck: Hermitian  eff. Hamiltonian (recommended)
                       # QD_Bloch   : non-Hermitian eff.
```

(continues on next page)

(continued from previous page)

```
                        # 0: disabled (default)

    # optional: compute the QD-Corrected density.
    QDPopulationAnalysis True # Repeat population analysis with the corrected␣
↪density
    QDNatOrbs            True # Get natural orbitals with the corrected density
  end
end
```

ORCA will print the eff. Hamiltonian matrix and its eigenvectors at the end of the calculation for `PrintLevel 4`.

```
=================================================================
                       QD-NEVPT2 Results
=================================================================
  *********************
   MULT 1,
  *********************

  Total Hamiltonian to be diagonalized
                  0          1
     0    -107.074594  -0.012574
     1      -0.011748-107.003810

  Right Eigenvectors
                  0          1
    0      -0.987232   0.170171
    1      -0.159292  -0.985414

--------------------------
        ROOT = 0
--------------------------
        Total Energy Correction : dE = -0.25309172934720
        Zero Order Energy       : E0 = -106.82353108218946
        Total Energy (E0+dE)    : E  = -107.07662281153667

--------------------------
        ROOT = 1
--------------------------
        Total Energy Correction : dE = -0.23103459727281
        Zero Order Energy       : E0 = -106.77074682157986
        Total Energy (E0+dE)    : E  = -107.00178141885267
```

By construction the Hamiltonian is non-Hermitian (`QDType QD_Bloch`). Hence, the computation of properties with the revised wave function, e.g., expectation values require left- and right eigenvectors. The diagonalization of the general matrices appearing in the Bloch formulations may occasionally lead to complex eigenpairs - an undesirable artifact. Although, the eigenvalues have typically only a small imaginary component, the results are not reliable and ORCA prints a warning.

```
--- complex eigenvalues and eigenvectors
  WARNING! QD-Matrix has eigenvalues with imaginary component! iE(0)=-0.000016
  WARNING! QD-Matrix has eigenvalues with imaginary component! iE(1)=0.000016
```

The `QD_VanVleck` option avoids the general eigenvalue decomposition. The equations are derived from second-order Van Vleck perturbation theory, which results in a Hermitian effective Hamiltonian.[473] The methodology is equivalent to the symmetrization of the Bloch Hamiltonian. The solution is always real and properties are easily accessible. Thus, **`QD_VanVleck` is the recommended approach in ORCA**. For a more detailed comparison of the different eff. Hamiltonian theories, we refer to the literature.[487, 488]
In all formulations, the energy denominator in the quasi-degenerate NEVPT2 is very sensitive to approximations. The canonicalization options with averaged orbitals and orbitals energies (`canonstep 0/2`) have the tendency to lessen the energy denominator. To avoid artifacts, the calculation is restricted to `canonstep 1`, where each state has its own set of orbitals.

If properties are requested within the casscf module, i.e. *zero-field splitting*, there will be an additional printing with the revised CI vectors and energies.

In addition, with `QDPopulationAnalysis true` the population analysis is repeated with the unrelaxed densities using the revised reference wave function. Using the directive `!KeepDens` these densities are stored and kept in the density container as "Tdens-CAS-DPDICPT2" together with a suffix containing the state details (multiplicity and root). Following the ORCA convention, the suffix ends with ".p" and ".r" for densities and spin-densities respectively. For clarity, we added a snapshot of the density container ("orca_plot jobname.densities"). Note that *properties*, such as *Mössbauer*, that are processed in `orca_prop`, will be re-computed with the revised densities, when it is available/allowed.

```
Index:                                                    Name of Density
-----------------------------------------------------------------------
    0:                                                          cas.scfp
    1:                              Tdens-CAS-DPDICPT2MO.mult.1.iroot.0.p
    2:                                Tdens-CAS-DPDICPT2.mult.1.iroot.0.p
    3:                              Tdens-CAS-DPDICPT2MO.mult.1.iroot.1.p
    4:                                Tdens-CAS-DPDICPT2.mult.1.iroot.1.p
    5:                                Tdens-CAS-DPDICPT2.mult.1.iroot.0.r
    6:                                Tdens-CAS-DPDICPT2.mult.1.iroot.1.r
```

The revised densities can also be used to form natural orbitals setting `QDNatorbs true`. The resulting orbitals are stored in a GBW file with the suffix "mult.A.iroot.B.QD-NEVPT2.natorbs", where "A" denotes the multiplicity and "B" the root count.

3.16.13 Keywords

NEVPT2 can also be set using the `PTMethod` keyword in the `%casscf` block. Beware of the underscore in the keywords using the block input. Refined settings are adjusted in the `PTSettings` subblock. The methods require higher order reduced density matrices, which are not implemented for all of the CISteps. In addition to the default CIStep, `ACCCI`, `ICE` and `DMRGCI` are supported.

```
%casscf

  # usual CASSCF settings defining the active space and states of interest
  ...
  MULT   3,1 # multiplicity block
  NRoots 2,3 # number of roots for the MULT blocks

  CIStep    ICE     # CSFCI  : default (recommended)
                    # ACCCI  : fast, but memory intense
                    # ICE    : larger active spaces with ICE FIC-NEVPT2
                    # DMRGCI : larger active spaces with DMRG SC-NEVPT2

  TrafoStep RI      # RI approximation for CASSCF and NEVPT2

  # calling the PT2 method of choice
  PTMethod    SC_NEVPT2    # strongly contracted NEVPT2
              FIC_NEVPT2   # fully internally contracted / partially contracted␣
↪NEVPT2
              DLPNO_NEVPT2 # FIC-NEVPT2 using the DLPNO framework for large␣
↪molecules

  # detailed settings (optional) for the PT2 approaches
  PTSettings
    PrintLevel 1      # default printing
    EWIN  -3,1000     # Energy window for the frozencore setting fc_ewin
    NThresh 1e-6      # FIC-NEVPT2 cut off for linear dependencies
```

(continues on next page)

(continued from previous page)

```
   # handling of reduced density matrices:
   D4Step  efficient # efficient: default 4-RDM handling
                     # fly      : memory saving algorithm for larger active spaces
                     # cu4 : cumulant approximation of the 4-RDM
                     # cu34: cumulant approximation of the 3-RDM and 4-RDM
   D4Tpre  1e-12     # PS approximation of the 4-pdm
   D3Tpre  1e-14     # PS approximation of the 3-pdm

   # false intruder state related options:
   TSMallDenom 1e-2 # printing thresh for small denominators
   MaxWarnings  5   # max number of warning before going silent
   imaginary    0.0 # imaginary shift for FIC-NEVPT2

   # option to skip the PT2 correction for all states except for the ones
   # specified for a selected multiplicity block and root
   # selectedRoots[block]  = root number counting from 0.
   selectedRoots[1]=0,1 # compute MULT=1 root=0,1 and skip all others.


   # SC-NEVPT2 specific features:
   QDType    QD_VanVleck      # QD_VanVleck: Van Vleck (recommended)
                              # QD_Bloch: Bloch Hamiltonian
                              # 0: disabled
   QDNatOrbs True             # get revised density and natural orbitals
   QDPopulationAnalysis True  # get revised density and repeat population analysis

   # SC-NEVPT2 state-average orbital canonicalization options:
   CanonStep   canonical # canonical for each state (default)
               averaged  # state-average
               relaxed   # canonical and partially relaxed for each state.
               0         # state-average, but state-specific orbital energies

   # canonstep=relaxed additional control parameters:
   GScaling 0.5          # damping factor for the stepsize
   GStep    NR           # relax with NR
            SOSCF        # relax with SOSCF
            SUPERCI_PT   # relax with perturbation based SuperCI

   # FIC-NEVPT2 specific features:
   F12       true      # add F12 correction  and CABSSingles correction for CASSCF
   Density   unrelaxed # unrelaxed density generated for each state.
   NatOrbs   true      # computes the natural orbitals

   # DLPNO specific settings for expert users (see DLPNO-NEVPT2 article):
   TCutPNO      1e-8 # cutoff for PNO occupation numbers
   TCutPNO_Core 1e-2 # cutoff for PNO occupation numbers for core orbitals
   TCutMKN      1e-3 # cutoff for Mulliken populations in the PNO integral␣
↪transformation
   TCutDO       1e-2 # cutoff for the sparse map construction of domain
   TCutDOij     1e-5 # cutoff for the differential overlap matrix of ij pairs in␣
↪the Prescreen step
   # DLPNO DIIS solver settings:
   MaxIter    30    # maximum number of iterations
   MaxDIIS    7     # DIIS max. dimension
   DIISDamp1  0.5   # damping of the residual before DIIS is switched ON
   DIISDamp2  0     # damping of the residual after DIIS is switched ON
   LevelShift 0.2   # LevelShift during update
   LMP2Damp   1.0   # damping for the LMP2 amplitude update
   LMP2TolE   1e-7  # energy convergence criteria
   LMP2TolR   5e-7  # residual convergence criteria
 end
```

(continues on next page)

(continued from previous page)

```
end
```

3.17 Complete Active Space Peturbation Theory (CASPT2 and CASPT2-K)

The fully internally contracted CASPT2 (FIC-CASPT2) approach is available with real, imaginary and IPEA shifts.[489, 490, 491]. The ORCA implementation employs a reformulation of the CASPT2, that completely avoids the fourth order reduced density matrix, that would appear in the canonical implementation.[479] Some concepts are shared by a recent development reported by Sokolov and coworkers.[480] The modification allows calculations with large active spaces without approximating the results e.g. with the cumulant expansion.

It should be noted that the IPEA shift in OpenMOLCAS slightly deviates from ORCA.[492]. Here, the IPEA shift, λ, is added to the matrix elements of the internally contracted CSFs $\Phi_{qs}^{pr} = E_q^p E_s^r|\Psi^0>$ with the generalized Fock operator

$$<\Phi_{q's'}^{p'r'}|\hat{F}|\Phi_{qs}^{pr}> += <\Phi_{q's'}^{p'r'}|\Phi_{qs}^{pr}> \cdot \frac{\lambda}{2} \cdot (4 + \gamma_p^p - \gamma_q^q + \gamma_r^r - \gamma_s^s),$$

where $\gamma_q^p =< \Psi^0|E_q^p|\Psi^0 >$ is the expectation value of the spin-traced excitation operator.[493] The labels p,q,r,s refer to general molecular orbitals (inactive, active and virtual). Irrespective of the ORCA implementation, the validity of the IPEA shift in general remains questionable and is thus by default disabled.[494] ORCA features an alternative formulation, named **CASPT2-K**, that revises the zeroth order Hamiltonian itself.[495] Here, two additional Fock matrices are introduced for excitation classes that add or remove electrons from the active space. The new Fock matrices are derived from the generalized Koopmans' matrices corresponding to electron ionization and attachment processes. The resulting method is less prone to intruder states and the same time more accurate compared to the canonical CASPT2 approach. For more a detailed discussion, we refer to the paper by Kollmar et al.[495]

CASPT2 and CASPT2-K belong to the same family of approaches as *NEVPT2*. Both methods are called in complete analogy to the FIC-NEVPT2 approach. Note that the methodology can be combined with the *RI approximation*. However, unlike NEVPT2, the CASPT2 formulations require the storage of the entire wave function. Hence, the scope of accessible systems is far more limited compared to the NEVPT2 implementation in ORCA. Below is concise *list with the accessible options*.

```
%casscf
  ...

  # mandatory
  PTMethod  FIC_CASPT2  # canonical CASPT2 approach
            FIC_CASPT2K # CASPT2-K with revised H0

  # optional
  TrafoStep RI      # optional for RI approximation for CASSCF and CASPT2
  PTSettings
    CASPT2_ishift 0.0    # imaginary level-shift
    CASPT2_rshift 0.0    # real      level-shift
    CASPT2_IPEAshift 0.0 # IPEA shift.
  end
end
```

CASPT2 can also be set using the simple keywords on top of any valid CASSCF input.

```
!CASPT2       # FIC-CASPT2
!CASPT2K      # FIC-CASPT2-K
!RI-CASPT2    # FIC-CASPT2 with RI approximation
!RI-CASPT2K   # FIC-CASPT2-K with RI approximation
%casscf ...
```

3.17.1 Example Nitrogen molecule

In this short section, we add the CASPT2 results to the previously computed *NEVPT2 potential energy surface* of the N_2 molecule.

```
! def2-svp nofrozencore
%casscf nel  6
        norb 6
        mult 1
        PTMethod FIC_CASPT2 # fully internally contracted CASPT2
end

# scanning from the outside to the inside
%paras
        R = 2.5,0.7, 30
end

*xyz 0 1
N 0.0 0.0 0.0
N 0.0 0.0 {R}
*
```

The CASPT2 output lists the settings prior to the computation. The printed reference weights should be checked. Small **reference weights** indicate intruder states. Along the lines, the program also prints the **smallest denominators** in the perturbation expression (highlighted in the snippet below). Small denominator may lead to intruder states.

```
------------------------------------------------------------
ORCA-CASSCF
------------------------------------------------------------

...
PT2-SETTINGS:
A PT2 calculation will be performed on top of the CASSCF wave function (PT2 =␣
↪CASPT2)
CASPT2 Real Levelshift            ...  0.00e+00
CASPT2 Im.  Levelshift            ...  0.00e+00
CASPT2 IPEA Levelshift            ...  0.00e+00
...
------------------------------------------------------------
                           < CASPT2  >
------------------------------------------------------------
...
-----------------------------------
CASPT2-D Energy =     -0.171839049
-----------------------------------

Class V0_ijab: dE=     -0.013891923
Class Vm1_iab: dE=     -0.034571085
Class Vm2_ab : dE=     -0.040985427
Class V1_ija : dE=     -0.003511548
Class V2_ij  : dE=     -0.000579508
Class V0_ia  : dE=     -0.075176596
Class Vm1_a  : dE=     -0.002917335
Class V1_i   : dE=     -0.000205627

smallest energy denominator IJAB =      3.237539973
smallest energy denominator ITAB =      2.500295823
smallest energy denominator IJTA =      2.339868413
smallest energy denominator TUAB =      1.664398302
smallest energy denominator IJTU =      1.342421639
smallest energy denominator ITAU =      1.496042538
smallest energy denominator TUVA =      0.706288250
```

(continues on next page)

(continued from previous page)

```
smallest energy denominator ITUV =      0.545304334

...
 Iter           EPT2     EHylleraas  residual norm    Time
    1    -0.17183905    -0.17057203     0.03246225     0.0
    2    -0.17057203    -0.17119523     0.00616509     0.0
    3    -0.17117095    -0.17121211     0.00086389     0.0
    4    -0.17120782    -0.17121281     0.00013292     0.0
    5    -0.17121273    -0.17121282     0.00000990     0.0
    6    -0.17121283    -0.17121282     0.00000159     0.0
    7    -0.17121282    -0.17121282     0.00000020     0.0

 CASPT2 calculation converged in 7 iterations

...
 ================================================================
 CASPT2 Results
 ================================================================
   *********************
   MULT 1, ROOT 0
   *********************

  Class V0_ijab :        dE = -0.013831560889
  Class Vm1_iab :        dE = -0.034124733943
  Class Vm2_ab  :        dE = -0.041334010085
  Class V1_ija  :        dE = -0.003446396316
  Class V2_ij   :        dE = -0.000584401134
  Class V0_ia   :        dE = -0.074688029120
  Class Vm1_a   :        dE = -0.002962355569
  Class V1_i    :        dE = -0.000241331405

 ----------------------------------------------------------------
         Total Energy Correction : dE = -0.17121281846205
 ----------------------------------------------------------------
         Reference  Energy       : E0 = -108.66619981448225
         Reference Weight        : W0 = 0.94765190644139
 ----------------------------------------------------------------
         Total Energy (E0+dE)    : E  = -108.83741263294431
 ----------------------------------------------------------------
```

Note that the program prints CASPT2-D results prior entering the CASPT2 iterations.[489] In case of intruder states, the residual equation may not converge. The program will not abort. Hence, it is important to check convergence for every CASPT2 run. In this particular example with the small basis sets, there are no intruder states.

Fig. 3.40: Potential Energy Surface of the N_2 molecule from CASSCF(6,6) and CASPT2 calculations (def2-SVP).

The potential energy surface in Fig. 3.40 is indeed very similar to the FIC-NEVPT2 approach, which is more efficient (no iterations) and robust (absence of intruder states). The figure also shows the CASPT2-K results, which is typically a compromise between the two methods. As expected, the largest deviation from CASPT2 is observed at the dissociation limit, where the open shell character dominates the reference wave function. In this example, the discrepancy between the three methods is rather subtle. However, the results may differ substantially on some challenging systems, such as Chromium dimer studied in the CASPT2-K publication. [495]. Despite its flaws, the CASPT2 method is of historical importance and remains a popular methodology. In the future we might consider further extension such as the (X)MS-CASPT2.[496]

3.17.2 Keywords

```
%casscf
  ...

  MULT   1,3 # multiplicity block
  NRoots 3,2 # number of roots for the MULT blocks

  TrafoStep RI      # optional for RI approximation for CASSCF and CASPT2
  PTMethod  FIC_CASPT2  # canonical CASPT2 approach
            FIC_CASPT2K # CASPT2-K with revised H0

  # detailed settings (this is optional)
  PTSettings
    CASPT2_ishift 0.0    # imaginary level-shift
    CASPT2_rshift 0.0    # real      level-shift
    CASPT2_IPEAshift 0.0 # IPEA shift.
```

(continues on next page)

(continued from previous page)

```
    MaxIter       20     # maximum for the CASPT2 iterations
    MaxDIIS       7      # maximum dimension of the DIIS expansion

    # intruder state warnings
    TSmallDenom   1e-2   # printing thresh for small denominators

    # general settings
    NThresh 1e-6  # FIC-CASPT2 cut off for linear dependencies
    D4Tpre  1e-12 # truncation of the 4-pdm
    D3Tpre  1e-14 # truncation of the 3-pdm
    EWIN  -3,1000 # Energy window for the frozencore setting fc_ewin

    # option to skip the PT2 correction for all states except for the ones
    # specified for a selected multiplicity block and root
    # selectedRoots[block]  = root number counting from 0.
    selectedRoots[0]=0,1 # compute MULT=1 root=0,1 and skip all others.

    #CASPT2-K specific options
    TReg 1e-2 # default for the Tikhonov reguralization
  end

end
```

3.18 CASSCF and DFT

The Complete Active Space Self-Consistent Field (*CASSCF*) method is effective at describing static correlation effects, which occur in cases such as open-shell molecules or the dissociation of covalent bonds. However, CASSCF is insufficient for describing dynamic correlation and is often too inaccurate for realistic predictions of energies and molecular properties.

Hybrid approaches that combine CASSCF with Density Functional Theory (*DFT*) have been developed as alternatives to traditional second-order perturbation theory methods like *NEVPT2* and *CASPT2*.

Since ORCA 6.1, the following hybrid CASSCF-DFT methods are available:

- Long-range CASSCF short-range DFT (*srDFT*)
- Multi-Configurational Pair Density Functional Theory (*MCPDFT*)

3.18.1 General Description

Multi-Configurational Pair Density Functional Theory

MCPDFT evaluates the Kohn-Sham DFT energy

$$\begin{aligned} E &= E^1 + E^{\mathrm{H}} + E^{\mathrm{xc}}[\rho, \Pi] \\ &= \sum_{pq} h_{pq} D_{pq} + \frac{1}{2} \sum_{pqrs} D_{pq} D_{rs} g_{pqrs} + E^{\mathrm{xc}}[\rho, \Pi] \end{aligned}$$

using the one-body density matrix (D_{pq}) and the two-body density matrix (d_{pqrs}), both obtained from a preceding CASSCF calculation.

To account for spin polarization in open-shell systems and to improve the accuracy in general, the exchange-correlation (xc) energy employs the on-top pair density $\Pi(\boldsymbol{r})$

$$\Pi = \sum_{pqrs} d_{pqrs}\, \phi_p(\boldsymbol{r})\, \phi_q(\boldsymbol{r})\, \phi_r(\boldsymbol{r})\, \phi_s(\boldsymbol{r})$$

This enables the computation of alpha and beta electron densities:

$$\rho_{\alpha/\beta} = \frac{1}{2}\left(\rho \pm \sqrt{\rho^2 - 2\Pi}\right) \tag{3.132}$$

For spatial points $\boldsymbol{r}$ where dynamic correlation is suppressed ($\rho^2 \geq 2\Pi$), well-established exchange-correlation functionals from single-reference DFT can be applied directly using Eq. (3.132). This technique is known as *functional translation*.[497]

In the original MCPDFT method,[497] regions where $\rho^2 < 2\Pi$ were approximated by setting $\rho_{\alpha/\beta} = \frac{1}{2}\rho$. MCPDFT functionals based on this approach are prefixed with `t`, e.g., `tLDA` for translated LDA.

However, in many systems, regions with $\rho^2 < 2\Pi$ contain important information and should not be neglected or approximated. Since $\rho_{\alpha/\beta}$ becomes complex in these regions, dedicated exchange-correlation functionals and their functional derivatives are required.[498, 499] Using Eq. (3.132) without approximation is called *complex functional translation*.[499] These functionals are prefixed with `ct`, e.g., `ctLDA`.

In ORCA 6.1 the folloing MCPDFT functionals are available:

Table 3.48: List of available MCPDFT functionals.

MCPDFT Functional	Description
tLDA	translated local density approximation (defaults to VWN5)
tPBE	translated Perdew, Becke, Ernzerhofer (PBE)
tPBE0	translated PBE0, 75 % PBE exchange, 25 % exact HF exchange
ctLDA	complex translated local density approximation (defaults to VWN5)
ctPBE	complex translated Perdew, Becke, Ernzerhofer (PBE)
ctPBE0	complex translated PBE0, 75 % PBE exchange, 25 % exact HF exchange

Note

The original MCPDFT is not variational. A variational form that minimizes the MCPDFT energy[500] is not yet available in ORCA.

For excited-state MCPDFT methods, see Sec. *Excited States*.

Long-Range CASSCF Short-Range DFT

Dynamic electron correlation is most significant when electrons are close together, making it a short-range phenomenon. The srDFT method[501] addresses this by applying DFT to the short-range part of the interaction and CASSCF to the long-range (static correlation) component.

The srDFT energy expression is:

$$E = \langle 0| \hat{H}^{\text{lr}} |0\rangle + E^{\text{sr-H}} + E^{\text{sr-xc}}$$

where $\hat{H}^{\text{lr}}$ is the long-range Hamiltonian evaluated with the CASSCF wavefunction $|0\rangle$, and $E^{\text{sr-H}}$ and $E^{\text{sr-xc}}$ are the short-range Hartree and exchange-correlation energies.

Range separation of explicit two-electron interactions for CASSCF and DFT affects two quantities:

- the **Coulomb operator** that is now split into a lr and sr part

$$\begin{aligned}\frac{1}{r_{12}} &= \hat{g}^{\text{lr}} + \hat{g}^{\text{sr}} \\ &= \frac{\text{erf}(\mu r_{12})}{r_{12}} + \frac{\text{erfc}(\mu r_{12})}{r_{12}}\end{aligned}$$

  leading to range-separated two-electron integrals

$$g^{\text{lr}}_{pqrs} = \int\int \phi_p(\boldsymbol{r}_1)\phi_q(\boldsymbol{r}_1)\,\hat{g}^{\text{lr}}\,\phi_r(\boldsymbol{r}_2)\phi_s(\boldsymbol{r}_2)\,d\boldsymbol{r}_1\,d\boldsymbol{r}_2$$

and g^{sr}_{pqrs}.

- dedicated **short-range xc functionals** in $E^{\text{sr-xc}}$ that are damped for an increasing fraction of CASSCF

An empirical **short-range damping parameter** μ controls the balance between the short-range and long-range mixing. Setting μ to 0 (zero) restores the single-reference Kohn-Sham DFT energy, as appropriate for a state-specific ground-state calculation. In contrast, increasing μ to very large values ($\mu > 100$) effectively recovers the CASSCF solution.

Unlike *MCPDFT*, srDFT is variationally minimized with respect to orbital and CI coefficients. Currently, this is only implemented for exchange-correlation functionals dependent on one-electron densities. For singlets, only total electron density ρ is used; for other spin multiplicities, both ρ, spin-density m, and their derivatives are considered.[502]

After convergence, the final srDFT energies can be re-evaluated using the on-top pair density as in *MCPDFT*

The following srDFT functionals are available in ORCA 6.1:

Table 3.49: List of available srDFT functionals.

MCPDFT Functional	Description
srLDA	short-range LDA (defaults to VWN5)[241, 503]
srPBE	short-range PBE, also known as Goll, Werner, Stoll (GWS) [504, 505]
srPBE0	short-range PBE0, 75 % srPBE exchange, 25 % exact sr HF exchange
sr-ctLDA	complex translated srLDA
sr-ctPBE	complex translated srPBE
sr-ctPBE0	complex translated srPBE0

3.18.2 Computational Aspects

In ORCA 6.1 the CASDFT methods *MCPDFT* and *srDFT* are implemented in the *TRAH-CASSCF* module. They can be invoked by adding the desired functional to the standard *CASSCF* input.

For example, a state-averaged CASDFT calculation on ethylene with π and π^* orbitals in the active space:

```
! TRAH
! ctPBE

%casscf
 nel    2
 norb   2
 mult   1
 nroots 3
end
```

```
! TRAH
! sr-ctPBE

%method
 RangeSepMu 0.40 # Default value
end

%casscf
 nel    2
 norb   2
 mult   1
 nroots 3
end
```

Note that the *RIJK* approximation is not yet available for CASDFT.

Note that translated sr functionals are also available. In that case, complex translation has to be switched off in the `%method` block

```
! TRAH
! sr-ctLDA

%method
 CASDFT
  ComplexTranslation false
 end
end
```

In contrast to single-reference *DFT*, the *RIJ* and *RIJCOSX* approximations are **not activated by default**. If desired, they must be explicitly given in the ORCA input file:

```
! TRAH
! ctPBE0
! RIJCOSX def2-TZVP/C DefGrid3

%casscf
 nel    2
 norb   2
 mult   3
 nroots 1
end
```

3.18.3 Excited States

Excited states can be calculated using state-averaged (SA) methods as in standard *CASSCF* calculations.

For MCPDFT, energies are evaluated from state-specific one- and two-body densities of a converged SA-CASSCF calculation.[497]

For srDFT, an approximate energy is first minimized using the averaged densities. Final srDFT energies are then evaluated from state-specific (SS) densities.[506]

(a) (a) (b) (b)

Fig. 3.41: Three lowest potential energy curves of ethylene twist calculated with various srLDA methods.

As shown in Fig. 3.41, srDFT with SS densities may produce false minima for twisted ethylene. MCPDFT suffers from similar issues.

To mitigate this, one can diagonalize a linearized CI-DFT Hamiltonian that accounts for deviations from averaged

densities:[506, 507]

$$
\begin{aligned}
E^{\text{CI-DFT}} \approx & \langle 0| \hat{h} |0\rangle + \delta_{1\,\text{srDFT}} \langle 0| \hat{g}^{\text{lr}} |0\rangle \\
& + E^{\text{(sr)H}}[\rho^{\text{SA}}] + E^{\text{(sr)xc}}[\rho^{\text{SA}}, \Pi^{\text{SA}}] \\
& + \int \left(\frac{\partial E^{\text{Hxc}}[\rho, \Pi]}{\partial \rho} \right)\bigg|_{*} \Delta\rho \, d\mathbf{r} + \int \left(\frac{\partial E^{\text{xc}}[\rho, \Pi]}{\partial \Pi} \right)\bigg|_{*} \Delta\Pi \, d\mathbf{r}
\end{aligned}
$$

As shown in Fig. 3.41, the CI-DFT variant (e.g., `sr-ctLDA`) resolves the false-minimum problem and yields the best agreement with the highly correlated *MRCI+Q* method.

The CI-DFT approach can be specified via the simple keyword input

```
! TRAH
! CI-sr-ctPBE

%casscf
 nel    2
 norb   2
 mult   1
 nroots 3
end
```

or via the method block

```
! TRAH
! sr-ctPBE

%method
 CASDFT
  OrthogonalCI true
 end
end

%casscf
 nel    2
 norb   2
 mult   1
 nroots 3
end
```

In ORCA 6.1 the following CI-DFT keywords are available:

Table 3.50: List of available CI-DFT keywords.

CI-DFT Keyword
CI-ctLDA
CI-ctPBE
CI-ctPBE0
CI-srLDA
CI-srPBE
CI-srPBE0
CI-sr-ctLDA
CI-sr-ctPBE
CI-sr-ctPBE0

3.19 Multireference Configuration Interaction and Pertubation Theory (uncontracted)

The multireference correlation module (`orca_mrci`) is designed for traditional (uncontracted) approaches (configuration interaction, MR-CI, and perturbation theory, MR-PT), which is the main focus of this section. For clarification, these approaches have in common that they consider excitations from each and every configuration state function (CSF) of the reference wavefunction. Hence, the computational cost of such approaches grows rapidly with the size of the reference space (e.g. CAS-CI). Furthermore, all integrals must be kept in memory to be efficient, which further limits the field of application.

Internally contracted multireference approaches, such as *N-electron Valence State Perturbation Theory (NEVPT2)*, do not share these bottlenecks. Here, excitations are defined with respect to the entire reference wavefunction. A number of internally contracted approaches (*NEVPT2*, *NEVPT4(SD)*, *FIC-MRCC* and *FIC-MRCI*) are available and described elsewhere in the manual.

> **Note**
>
> NEVPT2 is typically the method of choice as it is fast and easy to use. It is highly recommended to check the respective section, when new to the field. The following chapter focuses on the traditional multi-reference approaches as part of the `orca_mrci` module.

3.19.1 Introductory Remarks

All of the approaches in `orca_mrci`are uncontracted and start with a reference wavefunction that consists of multiple configurations (orbital occupation patterns). While the molecular orbitals typically arise from a CASSCF calculation, the reference wavefunction is not restricted to a complete active space (CAS). Restrictive active spaces (RAS) or arbitrary list of configurations can be defined in the `refs`sub-block.

The total wavefunction is constructed by considering single and double excitations out of the reference configurations. These excited configurations are then used to generate configuration state functions (CSF) that have the proper spin and spatial symmetry. The number of wavefunction parameters rapidly grows with the number of reference functions. The orca_mrci module features a set of truncation criteria (`TSel, TPre, TNat`) that help to reduce the number of wavefunction parameters. Furthermore, by default, the program only considers reference configurations that already have the target spin and spatial symmetry. There are situations, where this is undesired and the restrictions can be lifted with the keyword `rejectinvalidrefs false`. For more information on the theory, the program module as well as its usage we recommend the review article by Neese et al.[508]. A tutorial type introduction to the subject is presented in chapter *Multireference Configuration Interaction and Pertubation Theory (uncontracted)* of the manual and more examples in the CASSCF tutorial.

Although there has been quite a bit of experience with it, this part of the program is still somewhat hard to use and requires patience and careful testing before the results should be accepted. While we try to make your life as easy as possible, you have to be aware that ultimately any meaningful multireference *ab initio* calculation requires more insight and planning from the user side than standard SCF or DFT calculation or single reference correlation approaches like MP2 – so don't be fainthearted! You should also be aware that with multireference methods it is very easy to let a large computer run for a long time and still to not produce a meaningful result – your insight is a key ingredient to a successful application! Below a few examples illustrate some basic uses of the `orca_mrci` module.

3.19.2 RI-approximation

First of all, it is important to understand that the default mode of the MR-CI module in its present implementation performs a full integral transformation from the AO to the MO basis. This becomes very laborious and extremely memory intensive beyond approximately 200 MOs that are included in the CI. Alternatively, one can construct molecular electron-electron repulsion integrals from the resolution of the identity (RI) approximation. *Thus a meaningful auxiliary basis set must be provided if this option is chosen*. We recommend the fitting bases developed by the TurboMole developers for MP2 calculations. These give accurate transition energies; however, the error in the total energies is somewhat higher and may be on the order of 1 mEh or so. Check `IntMode` to change the default mode for the integral transformation. Note that in either way, the individually selecting MRCI module requires to have all integrals in memory which sets a limit on the size of the molecule that can be studied.

3.19.3 Individual Selection

Secondly, it is important to understand that the MR-CI module is of the *individually selecting* type. Thus, only those excited configuration state functions (CSFs) which interact more strongly than a given threshold ($\mathbf{T}_{\mathbf{sel}}$) with the 0th order approximations to the target states will be included in the variational procedure. The effect of the rejected CSFs is estimated using second order perturbation theory. The 0th order approximations to the target states are obtained from the diagonalization of the reference space configurations. A further approximation is to reduce the size of this reference space through another selection – all initial references which contribute less than a second threshold ($\mathbf{T}_{\mathbf{pre}}$) to the 0th order states are rejected from the reference space.

Single excitations

One important aspect concerns the single excitations. If the reference orbitals come from a CASSCF calculation the matrix elements between the reference state and the single excitations vanishes and the singles will not be selected. However, they contribute to fourth and higher orders in perturbation theory and may be important for obtaining smooth potential energy surfaces and accurate molecular properties. Hence, the default mode of the MRCI module requires to include all of the single excitations via the flag `AllSingles =true`. This may lead to lengthy computations if the reference spaces becomes large!

Reference Spaces

Third, the reference spaces in the MR-CI module can be of the complete active space (**CAS(n-electrons,m-orbitals)**) or restricted active space (**RAS**, explained later) type. It is important to understand that the program uses the orbitals around the HOMO-LUMO gap as provided by the user to build up the reference space! Thus, if the orbitals that you want to put in the active space are not coming "naturally" from your SCF calculation in the right place you have to reorder them using the "`moread`" and "`rotate`" features together with the `NoIter` directive. To select the most meaningful and economic reference space is the most important step in a multireference calculation. It *always* requires insight from the user side and also care and, perhaps, a little trial and error.

Size Consistency

Fourth, it is important to understand that CI type methods are *not* size consistent. Practically speaking the energy of the supermolecule A-B with noninteracting A and B fragments is not equal to the energies of isolated A and isolated B. There are approximate ways to account for this (**ACPF, AQCC** and **CEPA** methods) but the effect will be present in the energies, the more so the more electrons are included in the treatment. The same is *not* true for the perturbation theory based methods which are size consistent as long as the reference wavefunction is.

Performance

There are many flags that control the performance of the MR-CI program. Please refer to chapter 0 for a description of possible flags, thresholds and cut-offs. The most important thresholds are T_{sel} and T_{pre}, and for SORCI also T_{nat}.

Fig. 3.42: Zwitter-ionic form of Serine.

For some methods, like ACPF, it is possible to compare the performance of the MRCI module with the performance of the MDCI module. The MDCI module has been written to provide optimum performance if no approximations are introduced. The MRCI module has ben written more with the idea of flexibility rather than the idea of performance. Let us compare the performance of the two programs in a slightly nontrivial calculation – the zwitter-ionic form of serine. We compare the selecting MRCI approach with the approximation free MDCI module. The molecular size is such that still all four index integrals can be stored in memory.

Table 3.51: Comparison of the performance of the MRCI and MDCI modules for a single reference calculation with the bn-ANO-DZP basis set on the zwitter-ionic form of serine (14 atoms, 133 basis functions).

Module	Method	T_{sel}(Eh)	Time (sec)	Energy (Eh)
MRCI	ACPF	10^{-6}	3277	-397.943250
MDCI	ACPF	0	1530	-397.946429
MDCI	CCSD	0	2995	-397.934824
MDCI	CCSD(T)	0	5146	-397.974239

The selecting ACPF calculation selects about 15% of the possible double excitations and solves a secular problem of size ≈ 360,000 CSFs. The MDCI module ACPF calculation optimizes approximately 2.5 million wavefunction amplitudes — and this is not a large molecule or a large basis set! Despite the fact that the MDCI module makes no approximation, it runs twice as fast as the *selected* MRCI module and an estimated 50 times faster than the *unselected* MRCI module! This will become even more pronounced for the larger and more accurate basis sets that one should use in such calculations anyways. The error of the selection is on the order of 3 mEh or 2 kcal/mol in the total energy. One can hope that at least part of this error cancels upon taking energy differences.[1] The more rigorous CCSD

[1] Depending on whether one wants to take a pessimistic or an optimistic view one could either say that this result shows what can be achieved with a code that is dedicated to a single determinant reference. Alternatively one could (and perhaps should) complain about the high price one

calculation takes about a factor of two longer than the ACPF calculation which seems reasonable. The triples add another factor of roughly 2 in this example but this will increase for larger calculations since it has a steeper scaling with the system size. The ACPF energy is intermediate between CCSD and CCSD(T) which is typical — ACPF overshoots the effects of disconnected quadruples which partially compensates for the neglect of triples.

These timings will strongly depend on the system that you run the calculation on. Nevertheless, what you should take from this example are the message that if you can use the MDCI module, do it.

The MDCI module can avoid a full integral transformation for larger systems while the MRCI module can use selection and the RI approximation for larger systems. Both types of calculation will become very expensive very quickly! Approximate MDCI calculations are under development.

Symmetry

The MRCI program really takes advantage of symmetry adapted orbitals. In this case the MRCI matrix can be blocked according to irreducible representations and be diagonalized irrep by irrep. This is a big computational advantage and allows one to converge on specific excited states much more readily than if symmetry is not taken into account.

The syntax is relatively easy. If you specify:

```
newblock 1 *
        nroots 8
        refs cas(4,4) end
        end
```

Then the "*" indicates that this is to be repeated in each irrep of the point group. Thus, in C_{2v} the program would calculate 8 singlet roots in each of the four irreps of the C_{2v} point group thus leading to a total of 32 states.

Alternatively, you can calculate just a few roots in the desired irreps:

```
newblock 1 0
        nroots 3
        refs cas(4,4) end
        end
      newblock 1 2
        nroots 5
        refs cas(4,4) end
        end
      newblock 3 1
        nroots 1
        refs cas(4,4) end
        end
```

In this example, we would calculate 3 singlet roots in the irrep "0" (which is A_1), then five roots in irrep "2" (which is B_1) and then 1 triplet root in irrep 1 (which is B_2).

Obviously, the results with and without symmetry will differ slightly. This is due to the fact that without symmetry the reference space will contain references that belong to "wrong" symmetry but will carry with them excited configurations of "right" symmetry. Hence, the calculation without use of symmetry will have more selected CSFs and hence a slightly lower energy. This appears to be unavoidable. However, the effects should not be very large for well designed reference spaces since the additional CSFs do not belong to the first order interacing space.

pays for the generality of the MRCI approach. In any case, the name of the game would be to develop MR approaches that are equally efficient to single reference approaches. See FIC-MRCI chapter for more information.

3.19.4 A Tutorial Type Example of a MR Calculation

Perhaps, the most important use of the MR-CI module is for the calculation of transition energies and optical spectra. Let us first calculate the first excited singlet and triplet state of the formaldehyde molecule using the MR-CI method together with the Davidson correction to approximately account for the effect of unlinked quadruple substitutions. We deliberately choose a somewhat small basis set for this calculation which is already reasonable since we only look at a valence excited state and want to demonstrate the principle.

Suppose that we already know from a ground state calculation that the HOMO of H_2CO is an oxygen lone pair orbitals and the LUMO the π^* MO. Thus, we want to calculate the singlet and triplet n$\rightarrow$ π^* transitions and nothing else. Consequently, we only need to correlate two electrons in two orbitals suggesting a CAS(2,2) reference space.

```
# A simple MRCI example
! def2-SVP def2-SVP/C UseSym

%method frozencore fc_ewin
                 end

%mrci  ewin          -3,1000
       CIType           MRCI
       EUnselOpt     FullMP2
       DavidsonOpt Davidson1
       UseIVOs          true
       tsel             1e-6
       tpre             1e-2
       MaxMemInt         256
       MaxMemVec          32
       IntMode     FullTrafo
       AllSingles       true
       Solver           Diag
       # ground state 1A1
       NewBlock 1 0
         NRoots 1
         Excitations cisd
         Refs CAS(2,2) end
         End
       # HOMO LUMO transition 1A2
       NewBlock 1 1
         NRoots 1
         Excitations cisd
         Refs CAS(2,2) end
         End
       # HOMO LUMO triplet transition 3A2
       NewBlock 3 1
         NRoots 1
         Excitations cisd
         Refs CAS(2,2) end
         end
       end

* int 0 1
 C     0   0   0   0.000000     0.000     0.000
 O     1   0   0   1.200371     0.000     0.000
 H     1   2   0   1.107372   121.941     0.000
 H     1   2   3   1.107372   121.941   180.000
*
```

This input – which is much more than what is really required - needs some explanations: First of all, we choose a standard RHF calculation with the SVP basis set and we assign the SV/C fitting basis although it is not used in the SCF procedure at all. In the `%mrci` block all details of the MR-CI procedure are specified. First, `EWin` (`%method frozencore fc_ewin`) selects the MOs within the given orbital energy range to be included in the correlation treatment. The `CIType` variable selects the type of multireference treatment. Numerous choices are possible and

MRCI is just the one selected for this application.

> **Note**
>
> The CIType statement selects several default values for other variables. So it is a very good idea to place this statement at the beginning of the MR-CI block and possibly overwrite the program selected defaults later. If you place the `CIType` statement after one of the values which it selects by default your input will simply be overwritten!

The variables `EUnselOpt` and `DavidsonOpt` control the corrections to the MR-CI energies. `EUnselOpt` specifies the way in which the MR-CI energies are extrapolated to zero threshold T_{Sel}. Here we choose a full MR-MP2 calculation of the missing contributions to be done *after* the variational step, i.e. using the relaxed part of the reference wavefunction as a 0th order state for MR-PT. The `DavidsonOpt` controls the type of estimate made for the effect of higher substitutions. Again, multiple choices are possible but the most commonly used one (despite some real shortcomings) is certainly the choice `Davidson1`. The flag `UseIVOs` instructs the program to use "improved virtual orbitals". These are virtual orbitals obtained from a diagonalization of the Fock operator from which one electron has been removed in an averaged way from the valence orbitals. Thus, these orbitals "see" only a $N-1$ electron potential (as required) and are not as diffuse as the standard virtual orbitals from Hartree-Fock calculations. If you input DFT orbitals in the MR-CI moldule (which is perfectly admittable and also recommended in some cases, for example for transition metal complexes) then it is recommended to turn that flag off since the DFT orbitals are already o.k. in this respect. The two thresholds `Tsel` and `Tpre` are already explained above and represent the selection criteria for the first order interacting space and the reference space respectively. `Tsel` is given in units of Eh and refers to the second order MR-MP2 energy contribution from a given excited CSF. 10^{-6} Eh is a pretty good value. Reliable results for transition energies start with $\approx 10^{-5}$; however, the total energy is converging pretty slowly with this parameter and this is one of the greatest drawbacks of individually selecting CI procedures! (see below). `Tpre` is dimensionless and refers to the *weight* of a given initial reference after diagonalization of the given initial reference space (10^{-4} is a pretty good value and there is little need to go much lower. Aggressive values such as 10^{-2} only select the truly leading configurations for a given target state which can be time saving. Intermediate values are not really recommended). The parameters `MaxMemInt` and `MaxMemVec` tell the program how much memory (in MB) it is allowed to allocate for integrals and for trial and sigma-vectors respectively.

The flag `IntMode` tells the program to perform a full integral transformation. This is possible for small cases with less than, say, 100–200 MOs. In this case that it is possible it speeds up the calculations considerably. For larger molecules you *have to* set this flag to `RITrafo` which means that integrals are recomputed on the fly using the RI approximation which is more expensive but the only way to do the calculation. To switch between the possible modes use:

```
%mrci  IntMode FullTrafo # exact 4 index transformation
               RITrafo   # use auxiliary basis sets
```

For small molecules or if high accuracy in the total energies is required it is much better to use the exact four index transformation. The limitations are that you will run out of disk space or main memory with more than ca. 200–300 MOs.

The variable `Solver` can be `diag` (for Davidson type diagonalization) or `DIIS` for multirrot DIIS type treatments.

```
%mrci  Solver Diag   # Davidson solver
              DIIS   # Multiroot DIIS
```

For CI methods, the diag solver is usually preferable. For methods like ACPF that contain nonlinear terms, DIIS is imperative.

Next in the input comes the definition of what CI matrices are to be constructed and diagonalized. Each multiplicity defines a *block* of the CI matrix which is separately specified. Here we ask for two blocks – singlet and triplet. The general syntax is:

```
NewBlock Multiplicity Irrep
         NRoots 1           # Number of roots to determine
         Excitations cisd   # Type of excitations
```

(continues on next page)

(continued from previous page)

```
        Refs CAS(NEl,NOrb) end  # Reference space def.
        end                # Finalize the block
```

Now that all input is understood let us look at the outcome of this calculation:

The first thing that happens after the SCF calculation is the preparation of the frozen core Fock matrix and the improved virtual orbitals by the program `orca_ciprep`. From the output the energies of the IVOs can be seen. In this case the LUMO comes down to –8.2 eV which is much more reasonable than the SCF value of +3.... eV. Concomitantly, the shape of this MO will be much more realistic and this important since this orbital is in the reference space!

```
-----------------------------------------------------------------------------
                            ORCA CI-PREPARATION
-----------------------------------------------------------------------------

Reading the GBW file               ... done
Symmetry usage                     ... ON

One-Electron Matrix                ... test1.H.tmp
GBW-File                           ... test1.gbw
Improved virtual orbitals          ... test1.ivo
First MO in the CI                 ... 2
Internal Fock matrix               ... test1.fi.tmp
LastInternal Orbital               ... 6
Integral package used              ... LIBINT
Reading the GBW file               ... done
Symmetry usage                     ... ON

Reading the one-electron matrix    ... done
Forming inactive density           ... done
Forming averaged valence density   ...
Scaling the occupied orbital occupation numbers
First MO                            ... 2
Last MO                             ... 7
Number of electrons in the range    ...    12
Scaling factor                      ... 0.917

done
Forming internal density           ... done
Forming Fock matrix/matrices       ...
Nuclear repulsion    ...    31.371502
Core repulsion       ...    31.371502
One-electron energy  ...  -114.942080
Fock-energy          ...   -94.993431
Final value          ...   -73.596254
done
Modifying virtual orbitals         ...
Last occupied MO               ...    7
Total number of MOs            ...   38
Number of virtual MOs          ...   30
Doing diagonalization with symmetry
The improved virtual eigenvalues:
    0: -0.2955 au   -8.041 eV    2- B2
    1: -0.0701 au   -1.907 eV    6- A1
    2: -0.0176 au   -0.479 eV    3- B1
    3:  0.0064 au    0.175 eV    7- A1
    4:  0.2922 au    7.951 eV    8- A1
    5:  0.2948 au    8.021 eV    3- B2
    6:  0.3836 au   10.439 eV    4- B1
    7:  0.4333 au   11.790 eV    9- A1
    8:  0.4825 au   13.128 eV    5- B1
```

(continues on next page)

(continued from previous page)

```
    9:  0.5027 au   13.680 eV   10- A1
   10:  0.7218 au   19.642 eV   11- A1
   11:  0.8351 au   22.723 eV    4- B2
   12:  0.9371 au   25.501 eV    6- B1
   13:  1.0265 au   27.933 eV    1- A2
   14:  1.1141 au   30.317 eV   12- A1
   15:  1.2869 au   35.017 eV    5- B2
   16:  1.4605 au   39.743 eV    7- B1
...

done
Transforming integrals             ... done
Storing passive energy             ... done (    -73.59625384 Eh)
Transforming internal FI           ... done
             .... done with the Frozen Core Fock matrices
```

The next step is to transform the electron-electron repulsion integrals into the MO basis:

```
-------------------------
SHARK HALF TRANSFORMATION
-------------------------

Number of basis functions                ...     38
Number of operators                      ...      1
   Operator  0:    2-  37

Integral generator used                  ... SHARK
Contraction scheme used                  ... SEGMENTED CONTRACTION
MaxCore in resort                        ...    256 MB

Half transformed integrals for op=   0   ... test1.SHARK_MNPQ0.tmp
Resorted half transformed integrals      ... test1.JAO.tmp
Starting integral generation + half trafo...
Half trafo (segmented) done. Total time =  0.1 sec. integrals=  0.0 sec trafo=  0.
↪0 sec
Starting integral resorting              ... done (  0.0 sec)

SHARK half integral transformation done. Total time =  0.1 sec.


-------------------
FULL TRANSFORMATION
-------------------

Processing MO  10
Processing MO  20
Processing MO  30
Full transformation done
Number of integrals made            ...       222111
Number of integrals stored          ...        59070
Timings:
Time for first half transformation  ...     0.068 sec
Time for second half transformation ...     0.014 sec
Total time                          ...     0.086 sec
```

This will result in a few additional disk files required by `orca_mrci`. The program then tells you which multiplicities will be treated in this MRCI run:

```
------------------
CI-BLOCK STRUCTURE
------------------
```

(continues on next page)

(continued from previous page)

```
Number of CI-blocks                ... 3

===========
CI BLOCK 1
===========
Multiplicity             ...   1
Irrep                    ...   0
Number of reference defs ...   1
  Reference   1: CAS(2,2)

Excitation type          ... CISD
Excitation flags for singles:
    1 1 1 1
Excitation flags for doubles:
    1 1 1 / 1 1 1 / 1 1 1

===========
CI BLOCK 2
===========
Multiplicity             ...   1
Irrep                    ...   1
Number of reference defs ...   1
  Reference   1: CAS(2,2)

Excitation type          ... CISD
Excitation flags for singles:
    1 1 1 1
Excitation flags for doubles:
    1 1 1 / 1 1 1 / 1 1 1

===========
CI BLOCK 3
===========
Multiplicity             ...   3
Irrep                    ...   1
Number of reference defs ...   1
  Reference   1: CAS(2,2)

Excitation type          ... CISD
Excitation flags for singles:
    1 1 1 1
Excitation flags for doubles:
    1 1 1 / 1 1 1 / 1 1 1

        ---------------------------------------------------------------------
        -------------------- ALL SETUP TASKS ACCOMPLISHED ------------------
        --------------------      (     0.139 sec)        ------------------
        ---------------------------------------------------------------------
```

Now that all the setup tasks have been accomplished the MRCI calculation itself begins.

```
#####################################################
#                                                   #
#                       M R C I                     #
#                                                   #
# TSel    =   1.000e-06 Eh                          #
# TPre    =   1.000e-02                             #
# TIntCut =   1.000e-10 Eh                          #
# Extrapolation to unselected MR-CI by full MP2     #
# DAVIDSON-1 Correction to full CI                  #
```

(continues on next page)

(continued from previous page)

```
#                                                    #
#####################################################


---------------------
INTEGRAL ORGANIZATION
---------------------

Reading the one-Electron matrix             ... done
E0 read was -73.596253835266
Reading the internal Fock matrix            ... assuming it to be equal to the one-
↪electron matrix!!!
done
Preparing the integral list                 ... done
Loading the full integral                   ... done
Making the simple integrals                 ... done

                  ****************************************
                  *              CI-BLOCK  1             *
                  ****************************************

Configurations with insufficient # of SOMOs WILL be rejected
Building a CAS(2,2) for multiplicity 1 and irrep=A1
Reference Space:
 Initial Number of Configurations :      2
 Internal Orbitals :    2 -    6
 Active Orbitals   :    7 -    8
 External Orbitals :    9 -   37
The number of CSFs in the reference is 2
Calling MRPT_Selection with N(ref)=2
```

In the first step, the reference space is diagonalized. From this CI, the most important configurations are selected with `Tpre`:

```
------------------
REFERENCE SPACE CI
------------------

 Pre-diagonalization threshold                 : 1.000e-02
Warning: Setting NGuessMat to 512
N(ref-CFG)=2 N(ref-CSF)=2

                       ****Iteration    0****
   Lowest Energy          :  -113.779221580786
   Maximum Energy change  :   113.779221580786 (vector 0)
   Maximum residual norm  :     0.000000000000

      *** CONVERGENCE OF RESIDUAL NORM REACHED ***
Reference space selection using TPre= 1.00e-02

    ... found 1 reference configurations (1 CSFs)
        ... now redoing the reference space CI ...

Warning: Setting NGuessMat to 512
N(ref-CFG)=1 N(ref-CSF)=1

                       ****Iteration    0****
   Lowest Energy          :  -113.778810013503
   Maximum Energy change  :   113.778810013503 (vector 0)
   Maximum residual norm  :     0.000000000000
```

(continues on next page)

(continued from previous page)

```
      *** CONVERGENCE OF RESIDUAL NORM REACHED ***
```

In this case, the CAS space only has 2 correctly symmetry adapted CSFs one of which (the closed-shell determinant) is selected. In general, larger CAS spaces usually carry around a lot of unnecessary CSFs which are not needed for anything and then the selection is important to reduce the computational effort. The result of the second reference space CI is printed:

```
----------
CI-RESULTS
----------

The threshold for printing is  0.30 percent
The weights of configurations will be printed. The weights are summed over
all CSFs that belong to a given configuration before printing

STATE   0:  Energy=   -113.778810014 Eh RefWeight=  1.0000  0.00 eV      0.0 cm**-1
      1.0000  : h---h---[20]
```

Energy is the total energy in Eh. In the present case we can compare to the SCF energy `-113.778810014 Eh` and find that the reference space CI energy is identical, as it has to be since the lowest state coincides with the reference space. **RefWeight** gives the weight of the reference configurations in a CI state. This is 1.0 in the present case since there were only reference configurations. The number `1.000` is the weight of the following configuration in the CI vector. The description of the configuration **h–h–[20]p–p–** is understood as follows:[2] The occupation of the active orbitals is explicitly given in square brackets. Since the HOMO orbitals is number 7 from the SCF procedure, this refers to MOs 7 and 8 in the present example since we have two active orbitals. The `2` means doubly occupied, the `0` means empty. Any number (instead of –) appearing after an `h` gives the index of an internal orbital in which a hole is located. Simarly, any number after a `p` gives the index of an virtual (external) MO where a particle is located. Thus `h–h–[20]` is a closed shell configuration and it coincides with the SCF configuration—this was of course to be expected. The second root (in CI-Block 2) `h–h–[11]` by comparison refers to the configuration in which one electron has been promoted from the HOMO to the LUMO and is therefore the desired state that we wanted to calculate. Things are happy therefore and we can proceed to look at the output.

The next step is the generation of excited configurations and their selection based on `Tsel`:

```
-----------------------------
MR-PT SELECTION TSel= 1.00e-06
-----------------------------


Setting reference configurations WITH use of symmetry
Building active patterns WITH use of symmetry

Selection will be done from 1 spatial configurations
Selection will make use of spatial symmetry
 ( 0) Refs        : Sel:         1CFGs/         1CSFs Gen:           1CFGs/       ␣
↪   1CSFs  (       0.000 sec)
Building active space densities            ...        0.002 sec
Building active space Fock operators       ...        0.000 sec
 ( 1) (p,q)->(r,s): Sel:         1CFGs/         1CSFs Gen:           1CFGs/       ␣
↪   1CSFs  (       0.000 sec)
 ( 2) (i,-)->(p,-): Sel:         1CFGs/         1CSFs Gen:           1CFGs/       ␣
↪   1CSFs  (       0.000 sec)
 ( 3) (i,j)->(p,q): Sel:         8CFGs/         8CSFs Gen:           8CFGs/       ␣
↪   8CSFs  (       0.000 sec)
 ( 4) (i,p)->(q,r): Sel:         0CFGs/         0CSFs Gen:           1CFGs/       ␣
↪   1CSFs  (       0.000 sec)
 ( 5) (p,-)->(a,-): Sel:         8CFGs/         8CSFs Gen:           8CFGs/       ␣
↪   8CSFs  (       0.000 sec)
```

(continues on next page)

[2] Note that for printing we always sum over all linearly independent spin couplings of a given spatial configuration and only print the summed up weight for the configuration rather than for each individual CSF of the configuration.

(continued from previous page)

```
 ( 6) (i,-)->(a,-): Sel:        52CFGs/        52CSFs Gen:          52CFGs/
↪  52CSFs  (       0.000 sec)
 ( 7) (i,j)->(p,a): Sel:        95CFGs/       166CSFs Gen:          96CFGs/
↪ 167CSFs  (       0.000 sec)
 ( 8) (i,p)->(q,a): Sel:        21CFGs/        42CSFs Gen:          22CFGs/
↪  44CSFs  (       0.000 sec)
 ( 9) (p,q)->(r,a): Sel:         3CFGs/         3CSFs Gen:           5CFGs/
↪   5CSFs  (       0.000 sec)
 (10) (i,p)->(a,b): Sel:       555CFGs/      1082CSFs Gen:         584CFGs/
↪1139CSFs  (       0.001 sec)
 (11) (p,q)->(a,b): Sel:       124CFGs/       124CSFs Gen:         148CFGs/
↪ 148CSFs  (       0.000 sec)
 (12) (i,j)->(a,b): Sel:      1688CFGs/      2685CSFs Gen:        1887CFGs/
↪2947CSFs  (       0.001 sec)

Selection results:
Total number of generated configurations:       2814
Number of selected configurations       :       2557 ( 90.9%)
Total number of generated CSFs          :       4522
Number of selected CSFS                 :       4173 ( 92.3%)

The selected tree structure:
Number of selected Internal Portions       :     11
Number of selected Singly External Portions:     27
   average number of VMOs/Portion          :   6.39
   percentage of selected singly externals :  22.83
Number of selected Doubly External Portions:     21
   average number of VMOs/Portion          : 107.59
   percentage of selected doubly externals :  27.76
```

Here, the program loops through classes of excitations. For each excitation it produces the excited configurations (CFGs) and from it the linearly independent spin functions (CSFs) which are possible within the configuration. It then calculates the interaction with the contracted 0th order roots and includes all CSFs belonging to a given CFG in the variational space if the largest second order perturbation energy is larger or equal to `Tsel`. In the present case ≈136,000 CSFs are produced of which 25% are selected. For larger molecules and basis sets it is not uncommon to produce 10^9–10^{10} configurations and then there is no choice but to select a much smaller fraction than 20%. For your enjoyment, the program also prints the total energies of each state after selection:

```
Diagonal second order perturbation results:
State      E(tot)           E(0)+E(1)     E2(sel)      E2(unsel)
             Eh                Eh           Eh           Eh
-------------------------------------------------------------
 0    -114.108350270   -113.778810014    -0.329433    -0.000107
```

You can ignore this output if you want. In cases that the perturbation procedure is divergent (not that uncommon!) the total energies look strange—don't worry—the following variational calculation is still OK. The second order perturbation energy is here divided into a selected part `E2(sel)` and the part procedure by the unselected configurations `E2(unsel)`. Depending on the mode of `EUnselOpt` this value may already be used later as an estimate of the energetic contribution of the unselected CSFs.[3]

Now we have ≈ 4,200 CSFs in the variational space of CI block 1 and proceed to diagonalize the Hamiltonian over these CSFs using a Davidson or DIIS type procedure:

```
------------------------
DAVIDSON-DIAGONALIZATION
------------------------
```

(continues on next page)

[3] In this case the maximum overlap of the 0th order states with the final CI vectors is computed and the perturbation energy is added to the "most similar root". This is of course a rather crude approximation and a better choice is to recomputed the second order energy of the unselected configurations rigorously as is done with `EUnselOpt = FullMP2`.

(continued from previous page)

```
Dimension of the eigenvalue problem            ...   4173
Number of roots to be determined               ...      1
Maximum size of the expansion space            ...      4
Maximum number of iterations                   ...     35
Convergence tolerance for the residual         ...    1.000e-06
Convergence tolerance for the energies         ...    1.000e-06
Orthogonality tolerance                        ...    1.000e-14
Level Shift                                    ...    0.000e+00
Constructing the preconditioner                ... o.k.
Building the initial guess                     ... o.k.
Number of trial vectors determined             ...      4


                      ****Iteration    0****
   Size of expansion space: 3
   Lowest Energy          :  -113.937028067251
   Maximum Energy change  :   113.937028067251 (vector 0)
   Maximum residual norm  :     0.741727830968

                      ****Iteration    1****
   Size of expansion space: 4
   Lowest Energy          :  -114.082265676116
   Maximum Energy change  :     0.145237608865 (vector 0)
   Maximum residual norm  :     0.012707561344
   Rebuilding the expansion space

                      ****Iteration    2****
   Size of expansion space: 2
   Lowest Energy          :  -114.085350429118
   Maximum Energy change  :     0.003084753001 (vector 0)
   Maximum residual norm  :     0.002880697397

                      ****Iteration    3****
   Size of expansion space: 3
   Lowest Energy          :  -114.086043274125
   Maximum Energy change  :     0.000692845007 (vector 0)
   Maximum residual norm  :     0.000098595378

                      ****Iteration    4****
   Size of expansion space: 4
   Lowest Energy          :  -114.086074300143
   Maximum Energy change  :     0.000031026018 (vector 0)
   Maximum residual norm  :     0.000004959126
   Rebuilding the expansion space

                      ****Iteration    5****
   Size of expansion space: 2
   Lowest Energy          :  -114.086076038587
   Maximum Energy change  :     0.000001738444 (vector 0)
   Maximum residual norm  :     0.000000572348

      *** CONVERGENCE OF RESIDUAL NORM REACHED ***

Storing the converged CI vectors               ... test1.mrci.vec

                 *** DAVIDSON DONE ***
Returned from DIAG section
```

The procedure converges on all roots simultaneously and finishes after six iterations which is reasonable. Now the program calculates the Davidson correction (`DavidsonOpt`) which is printed for each root.

```
Davidson type correction:
Root= 0 W=   0.912 E0=  -113.778810014 ECI=  -114.086076039 DE=-0.026913
```

Already in this small example the correction is pretty large, ca. 27 mEh for the ground state (and ≈ 36 mEh for the excited state, later in the output). Thus, a contribution of ≈ 9 mEh = 0.25 eV is obtained for the transition energy which is certainly significant. Unfortunately, the correction becomes unreliable as the reference space weight drops or the number of correlated electrons becomes large. Here 0.912 and 0.888 are still OK and the system is small enough to expect good results from the Davidson correction.

The next step is to estimate the correction for the unselected configurations:

```
Unselected CSF estimate:
Full relaxed MR-MP2 calculation           ...

Selection will be done from 1 spatial configurations
Selection will make use of spatial symmetry
Selection will make use of spatial symmetry
Selection will make use of spatial symmetry
done
Selected MR-MP2 energies                  ...

 Root=  0  E(unsel)=     -0.000106931
```

In the present case this is below 1 mEh and also very similar for all three states such that it is not important for the transition energy.

```
----------
CI-RESULTS
----------

The threshold for printing is  0.30 percent
The weights of configurations will be printed. The weights are summed over
all CSFs that belong to a given configuration before printing

STATE   0:  Energy=   -114.113096211 Eh RefWeight=  0.9124  0.00 eV      0.0 cm**-1
      0.9124  : h---h---[20]
      0.0114  : h  6h  6[22]
```

The final ground state energy is `-114.113096211` which is an estimate of the full CI energy in this basis set. The leading configuration is still the closed-shell configuration with a weight of $\approx 91\%$. However, a double excitation *outside* the reference space contributes some 1%. This is the excitation MO6,MO6 →LUMO,LUMO. This indicates that more accurate results are expected once MO6 is also included in the reference space (this is the HOMO-1). The excited state is dominated by the HOMO-LUMO transition (as desired) but a few other single- and double- excitations also show up in the final CI vector.

Now that all CI vectors are known we can order the states according to increasing energy and print (vertical) transition energies:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is   -114.113096211 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    A1    0    0      0.000      0.000       0.0
  1    3    A2    0    2    134.086      3.649   29428.4
  2    1    A2    0    1    148.499      4.041   32591.8
```

This result is already pretty good and the transition energies are within ≈ 0.1 eV of their experimental gas phase values (≈ 3.50 and ≈ 4.00 eV) and may be compared to the CIS values of 3.8 and 4.6 eV which are considerably in error.

In the next step the densities and transition densities are evaluated and the absorption and CD spectra are calculated (in the dipole length formalism) for the spin-allowed transitions together with state dipole moments:

```
-----------------------------------------------------------------------------------
↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)    (nm)                 (au**2)    (au)    ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A1 ->  0-1A2   4.040866   32591.8   306.8   0.000000000   0.00000  -0.00000  ␣
↪0.00000   0.00000

-----------------------------------------------------------------------------------
↪-------
                     CD SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-------
     Transition      Energy     Energy  Wavelength    R        MX        MY       ␣
↪MZ
                      (eV)      (cm-1)    (nm)   (1e40*cgs)   (au)      (au)     ␣
↪(au)
-----------------------------------------------------------------------------------
↪-------
  0-1A1 ->  0-1A2   4.040866   32591.8   306.8   -0.00000  -0.00000  -0.00000   0.
↪59348


-----------------------------------------------------------------------------
                            STATE DIPOLE MOMENTS
-----------------------------------------------------------------------------
  Root  Block           TX        TY        TZ           |T|
                      (Debye)   (Debye)   (Debye)      (Debye)
-----------------------------------------------------------------------------
    0    0            0.00000  -0.00000   2.33244      2.33244
    0    2            0.00000  -0.00000   1.45831      1.45831
    0    1            0.00000  -0.00000   1.58658      1.58658
```

Here the transition is symmetry forbidden and therefore has no oscillator strength. The state dipole moment for the ground state is 2.33 Debye which is somewhat lower than 2.87 Debye from the SCF calculation. Thus, the effect of correlation is to reduce the polarity consistent with the interpretation that the ionicity of the bonds, which is always overestimated by HF theory, is reduced by the correlation. Finally, you also get a detailed population analysis for each generated state density which may be compared to the corresponding SCF analysis in the preceding part of the output.

This concludes the initial example on the use of the MR-CI module. The module leaves several files on disk most of which are not yet needed but in the future will allow more analysis and restart and the like. The `.ivo` file is a standard `.gbw` type file and the orbitals therein can be used for visualization. This is important in order to figure out the identity of the generated IVOs. Perhaps they are not the ones you wanted and then you need to re-run the MR-CI with the IVOs as input, `NoIter` and the IVO feature in the new run turned off! We could use the IVOs as input for a state averaged CASSCF calculation:

```
! moread UseSym KDIIS
%moinp "Test-SYM-MRCI-H2CO.ivo"

%casscf nel    2
        norb   2
        irrep  0,1,1
```

(continues on next page)

(continued from previous page)

```
        mult   1,1,3
        nroots 1,1,1
        end
```

If we based a MR-ACPF calculation on this reference space we will find that the calculated transition energies are slightly poorer than in the MRCI+Q calculation. This is typical of approximate cluster methods that usually require somewhat larger reference spaces for accurate results. A similar result is obtained with SORCI.

```
%mrci  CIType          SORCI
        tsel             1e-6
        tpre             1e-4
        tnat             1e-5
        AllSingles       true
        doNatOrbs        true
        IntMode     FullTrafo
        # ground state 1A1
        NewBlock 1 0
          NRoots 1
          Excitations cisd
          Refs CAS(2,2) end
          End
        # HOMO LUMO transition 1A2
        NewBlock 1 1
          NRoots 1
          Excitations cisd
          Refs CAS(2,2) end
          End
        # HOMO LUMO triplet transition 3A2
        NewBlock 3 1
          NRoots 1
          Excitations cisd
          Refs CAS(2,2) end
          End
end
```

This gives:

```
State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    A1    0    0      0.000      0.000       0.0
  1    3    A2    0    2    144.563      3.934   31728.0
  2    1    A2    0    1    161.179      4.386   35374.7
```

This is systematically 0.4 eV too high. But let us look at the approximate average natural orbital (AANOs) occupation numbers:

```
------------------------
AVERAGE NATURAL ORBITALS
------------------------

Trace of the density to be diagonalized = 12.000000
Sum of eigenvalues = 12.000000
Natural Orbital Occupation Numbers:
N[  2] ( A1)=   1.99831062
N[  3] ( A1)=   1.99761604
N[  4] ( A1)=   1.99479313
N[  5] ( B1)=   1.99016881
N[  6] ( B2)=   1.95818285
N[  7] ( B1)=   1.33014178
N[  8] ( B2)=   0.70688423
N[  9] ( B1)=   0.00988561
N[ 10] ( A1)=   0.00436843
```

This shows that there is a low-occupancy orbital (MO6) that has not been part of the reference space. Thus, we try the same calculation again but now with one more active orbital and two more active electrons:

```
! moread
%moinp "Test-SYM-MRCI-H2CO.gbw"

%casscf nel 4
        norb 3
        irrep  0,1,1
        mult   1,1,3
        nroots 1,1,1
        end

%mrci CIType          SORCI
      tsel             1e-6
      tpre             1e-4
      tnat             1e-5
      AllSingles       true
      doNatOrbs        true
      IntMode     FullTrafo
      # ground state 1A1
      NewBlock 1 0
        NRoots 1
        Excitations cisd
        Refs CAS(4,3) end
        End
      # HOMO LUMO transition 1A2
      NewBlock 1 1
        NRoots 1
        Excitations cisd
        Refs CAS(4,3) end
        End
      # HOMO LUMO triplet transition 3A2
      NewBlock 3 1
        NRoots 1
        Excitations cisd
        Refs CAS(4,3) end
        End
end
```

This gives:

```
State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    A1    0    0      0.000      0.000       0.0
  1    3    A2    0    2    145.494      3.959   31932.3
  2    1    A2    0    1    162.222      4.414   35603.6
```

Which is now fine since all essential physics has been in the reference space. Inspection of the occupation numbers show that there is no suspicious orbital any more. Note that this is still a much more compact calculation that the MRCI+Q.

Likewise, we get an accurate result from MRACPF with the extended reference space.

```
State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    A1    0    0      0.000      0.000       0.0
  1    3    A2    0    2    134.985      3.673   29625.8
  2    1    A2    0    1    148.330      4.036   32554.6
```

However, the SORCI calculation is *much* more compact. For larger molecules the difference becomes more and more pronounced and SORCI or even MRDDCI2 (with or without +Q) maybe the only feasible methods—if at all.

3.19.5 Excitation Energies between Different Multiplicities

As an example for a relatively accurate MRCI+Q calculation consider the following job which calculates the triplet-ground and as the first excited singlet states of O_2.

```
! ano-pVQZ RI-AO cc-pVQZ/JK VeryTightSCF NoPop Conv UseSym RI-MP2 PModel
%mp2 density relaxed natorbs true end
%base "O2"
* xyz 0 3
O  0 0 0
O  0 0 1.2
*

$new_job
! ano-pVQZ RI-AO cc-pVQZ/JK VeryTightSCF NoPop Conv UseSym KDIIS
! moread
%moinp "O2.mp2nat"
%casscf nel     8
        norb    6
        irrep   1,0,1
        nroots  1,2,1
        mult    3,1,1
        trafostep    ri
        switchstep   nr
        end

%mrci   citype mrci
        tsel 1e-7
        tpre 1e-5
        newblock 3 1 nroots 1 refs cas(8,6) end end
        newblock 1 0 nroots 2 refs cas(8,6) end end
        newblock 1 1 nroots 1 refs cas(8,6) end end
        end

* xyz 0 3
O  0 0 0
O  0 0 1.2
*
```

Note that the linear molecule is run in D_{2h}. This creates a slight problem as the CASSCF procedure necessarily breaks the symmetry of the $^1\Delta$ state.

```
LOWEST ROOT (ROOT 0, MULT 3, IRREP B1g) =   -149.765383866 Eh -4075.323 eV

STATE   ROOT MULT IRREP DE/a.u.    DE/eV    DE/cm**-1
   1:    0    1  B1g   0.033334     0.907   7316.0
   2:    0    1   Ag   0.033650     0.916   7385.3
   3:    1    1   Ag   0.062381     1.697  13691.1
```

The result of the MRCI+Q is:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is   -150.176905551 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    3   B1g    0    0      0.000      0.000       0.0
  1    1   B1g    0    2     36.971      1.006    8114.2
  2    1    Ag    0    1     38.021      1.035    8344.7
  3    1    Ag    1    1     62.765      1.708   13775.2
```

These excitation energies are accurate to within a few hundred wavenumbers. Note that the ≈ 200 wavenumber splitting in the degenerate $^1\Delta$ state is due to the symmetry breaking of the CAS and the individual selection. Repeating the calculation with the MP2 natural orbitals gives an almost indistinguishable result and a ground state energy that is even lower than what was found with the CASSCF orbitals. Thus, such natural orbitals (that might often be easier to get) are a good substitute for CASSCF orbitals and at the same time the symmetry breaking due to the use of symmetry appears to be difficult to avoid.

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is   -150.177743426 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    3   B1g    0    0      0.000      0.000       0.0
  1    1   B1g    0    2     37.369      1.017    8201.5
  2    1    Ag    0    1     38.237      1.040    8392.1
  3    1    Ag    1    1     62.731      1.707   13767.9
```

3.19.6 Correlation Energies

The logic we are following here is the following: CID minus SCF gives the effect of the doubles; going to CISD gives the effect of the singles; QCISD(=CCD) minus CID gives the effect of the disconnected quadruples. QCISD minus QCID gives simultaneously the effect of the singles and the disconnected triples. They are a bit difficult to separate but if one looks at the singles alone and compares with singles + disconnected triples, a fair estimate is probably obtained. Finally, QCISD(T) minus QCISD gives the effect of the connected triples. One could of course also use CCSD instead of QCISD but I felt that the higher powers of T_1 obscure the picture a little bit—but this is open to discussion of course.

First H_2O/TZVPP at its MP2/TZVPP equilibrium geometry ($T_{\text{pre}} = 10^{-6}$ and $T_{\text{sel}} = 10^{-9}$ Eh for the MRCI and MRACPF calculations):

Excitation class	Energy (Eh)	Delta-Energy (mEh)
None (RHF)	-76.0624	
Doubles (CID)	-76.3174	255
+Singles (CISD)	-76.3186	1
+Disconnected Quadruples (QCID)	-76.3282	11
+Disconnected Triples (QCISD)	-76.3298	2
+Connected Triples (QCISD(T))	**-76.3372**	7
CASSCF(8,6)	-76.1160	
CASSCF(8,6) + MRCI	-76.3264	210
CASSCF(8,6) + MRCI+Q	**-76.3359**	10
CASSCF(8,6) + MRACPF	**-76.3341**	218

One observes quite good agreement between single- and multireference approaches. In particular, the contribution of the disconnected triples and singles is very small. The estimate for the disconnected quadruples is fairly good from either the multireference Davidson correction or the ACPF and the agreement between CCSD(T) and these MR methods is 2-3 mEh in the total energy which is roughly within chemical accuracy.

In order to also have an open-shell molecule let us look at NH with a N-H distance of 1.0 Å using the TZVPP basis set.

Excitation class	Energy (Eh)	Delta-Energy (mEh)
None (UHF)	-54.9835	
Doubles (CID)	-55.1333	150
+Singles (CISD)	-55.1344	1
+Disconnected Quadruples (QCID)	-55.1366	3
+Disconnected Triples (QCISD)	-55.1378	1
+Connected Triples (QCISD(T))	**-55.1414**	4
CASSCF(6,5)	-55.0004	
CASSCF(6,5) + MRCI	-55.1373	137
CASSCF(6,5) + MRCI+Q	**-55.1429**	6
CASSCF(6,5) + MRACPF	**-55.1413**	141

Again, the agreement is fairly good and show that both single- and multiple reference approaches converge to the same limit.

3.19.7 Thresholds

Now we choose the CO molecule (1.128 Ångström) with the SVP basis set and study the convergence of the results with respect to the selection threshold. Comparison to high level single-reference approaches is feasible (The SCF energy is `-112.645 946 Eh`).

Reference Values for Total Energies

The single-reference values are:

```
BD: -112.938 48002
              CCSD: -112.939 79145
             QCISD: -112.941 95700
             BD(T): -112.950 17278
           CCSD(T): -112.950 63889
          QCISD(T): -112.951 37425
         MP4(SDTQ): -112.954 80113
```

The calculations without connected triples (BD, CCSD, QCISD) are about the best what can be achieved without explicitly considering triple excitations. The CCSD is probably the best in this class. As soon as connected triples are included the CCSD(T), QCISD(T) and BD(T) values are close and from experience they are also close to the full CI values which is then expected somewhere between –112.950 and –112.952 Eh.

Convergence of Single Reference Approaches with Respect to T_{sel}

Next it is studied how these single reference methods converge with T_{sel}:

```
Closed-Shell ACPF:
   Tsel        Energy       (NCSF)    Energy       (NCSF)
   (Eh)       AllSingles=true      AllSingles=false
  TSel=0      -112.943 387 (5671)
  TSel=1e-14  -112.943 387 (2543)  -112.943 387 (2478)
  TSel=1e-10  -112.943 387 (2543)  -112.941 023 (2453)
  TSel=1e-08  -112.943 387 (2451)  -112.937 087 (2346)
  TSel=1e-06  -112.943 350 (2283)  -112.937 046 (2178)
  TSel=1e-05  -112.943 176 (1660)  -112.936 821 (1555)
  TSel=1e-04  -112.944 039 ( 782)  -112.938 381 ( 677)
```

It is clear that the convergence is erratic if the singles are not automatically included. This is the reason for making this the default from release 2.6.35 on. In the present case singles will only be selected due to round-off errors since by Brillouin's theorem the singles have zero-interaction with the ground state determinant. Thus, for individually selecting single-reference methods it is a good idea to automatically include all single-excitations in order to get converged results. The alternative would be a different singles selection procedure which has not yet been developed however. The selection of doubles appear to converge the total energies reasonably well. It is seen that the selection selects most CSFs between 10^{-5} and 10^{-7} Eh. Already a threshold of 10^{-6} Eh yields an error of less than 0.1 mEh which is negligible in relation to reaction energies and the like. Even 10^{-5} Eh gives an error of less than 0.1 kcal/mol.

Convergence of Multireference Approaches with Respect to T_{pre}

We next turn to multireference treatments. Here we want to correlate all valence electrons in all valence orbitals and therefore a CAS(10,8) is the appropriate choice. We first ask for the converged value of T_{pre} by using T_{sel} $=10^{-14}$ and obtain for MRCI+Q:

```
TPre = 1e-1: -112.943 964
         1e-2: -112.952 963
         1e-3: -112.953 786
         1e-4: -112.954 019
         1e-5: -112.954 336
         1e-6: -112.954 416
         1e-7: -112.954 440
```

Thus, pretty good convergence is obtained for $T_{pre} = 10^{-4} - 10^{-6}$. Hence 10^{-4} is the default.

To show a convenient input consider the following:

```
#
# Here we calculate the CO ground state correlation energy with several methods
#
! Def2-SVP Def2-SV/C RI-MP2 CCSD(T)
%base "1"

%mp2  density relaxed
      donatorbs  true
      end

* int 0 1
 C     0   0   0   0.000000     0.000     0.000
 O     1   0   0   1.128        0.000     0.000
*

$new_job

! aug-SVP MRACPF
! moread
%moinp "1.mp2nat"
# the CASSCF is done with MP2 natural orbitals which is a good idea and
# secondly we use a large level shift in order to help convergence
%casscf nel     10
        norb     8
        mult     1
        nroots   1
        shiftup  2
        shiftdn  2
        end

%mrci   tsel 1e-8
        tpre 1e-6
        end
```

(continues on next page)

(continued from previous page)

```
* int 0 1
 C     0   0   0   0.000  0.000  0.000
 O     1   0   0   1.128  0.000  0.000
*
```

This job computes at the same time all of the below and demonstrates once more the agreement between consequent single- and multireference correlation methods

```
SCF           =  -112.6459
RI-MP2        =  -112.9330
CCSD          =  -112.9398
CCSD(T)       =  -112.9506
CASSCF(10,8)  =  -112.7769
MRACPF        =  -112.9514
```

3.19.8 Energy Differences - Bond Breaking

For the calculation of energy differences we start again with the reference CCSD(T) calculation; this method is one of the few which can claim chemical accuracy in practical applications:

```
Reference Total Energies for N2 at 1.0977 Angstr\"{o}m with
  The SVP basis
    E(CCSD)   = -109.163 497
    E(CCSD(T))= -109.175 625
  Nitrogen Atom (4S), SVP basis, unrestricted
    E(CCSD)   =  -54.421 004
    E(CCSD(T))=  -54.421 7183
  Energy Difference:
  Delta-E(CCSD)   = -0.321 489 = 8.75 eV
  Delta-E(CCSD(T))= -0.332 188 = 9.04 eV
```

The basis set is of course not suitable for quantitative comparison to experimental values. However, this is not the point here in these calculations which are illustrative in nature. The SVP basis is just good enough to allow for a method assessment without leading to excessively expensive calculations.

This is now to be compared with the corresponding energy differences computed with some single-reference approaches. A typical input is (this is a somewhat old-fashioned example – in the present program version you would do a full valence CASSCF(10,8) or CASSCF(6,6) and invoke the MR-methods with a single keyword):

```
! HF def2-SVP def2-TZVPP/C VeryTightSCF NoPop

%base "1"

* xyz 0 1
N 0 0 0
N 0 0 1.0977
*
%method
     frozencore fc_ewin
end

%mrci
     EWin          -3,1000
     CIType       MRACPF2a
     Solver           DIIS
     IntMode      FullTrafo
     UseIVOs          true
     AllSingles       true
     TSel            1e-14
```

(continues on next page)

(continued from previous page)

```
     TPre            1e-05
     TNat              0.0
     ETol            1e-10
     RTol            1e-10
     NewBlock 1 *
          NRoots 1
          Excitations CISD
          refs CAS(0,0) end
     end
end

$new_job

! ROHF def2-SVP def2-TZVPP/C VeryTightSCF NoPop PModel

%base "2"

* xyz 0 4
N 0 0 0
*
%method
       frozencore fc_ewin
end

%mrci
      EWin          -3,1000
      CIType       MRACPF2a
      IntMode      FullTrafo
      UseIVOs          true
      AllSingles       true
      TSel            1e-14
      TPre            1e-05
      TNat              0.0
      ETol            1e-10
      RTol            1e-10
      NewBlock 4 *
          NRoots 1
          Excitations CISD
          refs CAS(3,3) end
      end
end
```

The results are:

```
Single reference approaches:
  Method   N2-Molecule         N-Atom        Delta-E
  CISD+Q : -109.167 904        -54.422 769    8.77 eV
  ACPF   : -109.166 926        -54.421 783    8.80 eV
  ACPF2  : -109.166 751        -54.421 333    8.82 eV
  ACPF2a : -109.166 730        -54.421 186    8.83 eV
  CEPA1  : -109.159 721        -54.422 564    8.56 eV
  CEPA2  : -109.172 888        -54.422 732    8.91 eV
  CEPA3  : -109.161 034        -54.422 589    8.59 eV
  AQCC   : -109.160 574        -54.420 948    8.67 eV
  CEPA-0 : -109.174 924        -54.422 951    8.95 eV
```

With exception is CEPA1 and CEPA3, the results are OK. The reason for the poor performance of these methods is simply that the formalism implemented is only correct for closed shells – open shells require a different formalism which we do not have available in the MRCI module (but in the single reference MDCI module). Due to the simple approximations made in CEPA2 it should also be valid for open shells and the numerical results are in support of that.

Next we turn to the multireference methods and take a CAS(10,8) reference as for CO in order to correlate all valence

electrons.[4]

```
Multi reference approaches:
  Method     N2-Molecule         N-Atom        Delta-E
  MRCISD+Q: -109.180 089        -54.422 667    9.11 eV
  MRACPF  : -109.178 708        -54.421 685    9.12 eV
  MRACPF2 : -109.177 140        -54.421 236    9.11 eV
  MRAQCC  : -109.175 947        -54.420 851    9.10 eV
  SORCI   : -109.179 101        -54.422 703    9.08 eV
```

This test calculation pleasingly shows the high consistency of multireference approaches which all converge more or less to the same result which must be accurate.

3.19.9 Energy Differences - Spin Flipping

There are a number of interesting situations in which one is interested in a small energy difference which arises from two states of different multiplicity but same orbital configuration. This is the phenomenon met in diradicals or in magnetic coupling in transition metal complexes. As a primitive model for such cases one may consider the hypothetical molecule H-Ne-H in a linear configuration which will be used as a model in this section.

The reference value is obtained by a MR-ACPF calculation with all valence electrons active (again, this example is somewhat old fashioned – in the present program version you would do a CASSCF calculation followed by MR methods with a single keyword):

```
! ROHF def2-SVP def2-TZVPP/C VeryTightSCF NoPop
%basis
     NewAuxCGTO Ne "AutoAux" end
end
* xyz 0 3
H  0 0 0
Ne 0 0 2.0
H  0 0 4.0
*
%method frozencore fc_ewin
        end

%mrci  EWin       -3,1000
       CIType     MRACPF2a
       IntMode   FullTrafo
       Solver         DIIS
       UseIVOs        true
       TSel              0
       TPre          1e-10
       ETol          1e-09
       RTol          1e-09
       DoDDCIMP2      true
       NewBlock 1 *
          NRoots 1
          Excitations CISD
          refs CAS(10,6) end
          end
       NewBlock 3 *
          NRoots 1
          Excitations CISD
          refs CAS(10,6) end
          end
       end
```

[4] Most of these results have been obtained with a slightly earlier version for which the MR energies are a little different from that what the present version gives. The energy differences will not be affected.

which gives the reference value 108 cm^{-1}. We now compare that to several other methods which only have the two "magnetic" orbitals (the 1s's on the hydrogens) in the active space:

```
... same as above
%mrci  EWin       -10,1000
       CIType      MRDDCI3
       ... same as previously
       NewBlock 1 *
          NRoots 1
          refs CAS(2,2) end
          end
       NewBlock 3 *
          NRoots 1
          refs CAS(2,2) end
          end
       end
```

This gives the result:

```
Method      S-T gap
MR-CI+Q  :  98 cm-1
MR-CI    :  93 cm-1
MR-ACPF  :  98 cm-1
MR-ACPF2 :  98 cm-1
MR-ACPF2a:  97 cm-1
MR-AQCC  :  95 cm-1
SORCI    : 131 cm-1
MR-DDCI2 :  85 cm-1
MR-DDCI3 : 130 cm-1
```

All these methods give good results with SORCI leading to a somewhat larger error than the others. The (difference dedicated CI) DDCI2 method slightly underestimates the coupling which is characteristic of this method. It is nice in a way that DDCI3 gives the same result as SORCI since SORCI is supposed to approximate the DDCI3 (or better the IDDCI3) result which it obviously does.

This splitting can also be studied using broken symmetry HF and DFT methods as explained elsewhere in this manual:

```
Method    S-T gap
 UHF     : 70 cm-1
 B3LYP/G : 240 cm-1
 BP86    : 354 cm-1
 PW91    : 234 cm-1
 PBE     : 234 cm-1
 PBE0    : 162 cm-1
 RPBE    : 242 cm-1
```

This confirms the usual notions; UHF underestimates the coupling and DFT overestimates it, less so for hybrid functionals than for GGAs. The BP86 is worse than PW91 or PBE. The PBE0 hybrid may be the best of the DFT methods. For some reason most of the DFT methods give the best results if the BS state is simply taken as an approximation for the true open-shell singlet. This is, in our opinion, not backed up by theory but has been observed by other authors too.

Now let us study the dependence on T_{sel} as this is supposed to be critical (we use the DDCI3 method):

```
Tsel     S-T gap
    1e-04     121
    1e-05     128
    1e-06     132
    1e-07     131
    1e-08     131
    1e-10     131
    1e-12     131
    0         131
```

The convergence is excellent once `AllSingles` are included.

3.19.10 Potential Energy Surfaces

Another situation where multireference approaches are necessary is when bond breaking is studied and one wants to calculate a full potential energy surface. Say we want to compute the potential energy surface of the CH molecule. First we have to figure out which states to include. Hence, let us first determine a significant number of roots for the full valence CASSCF reference state (we use a small basis set in order to make the job fast).

```
! ANO-pVDZ VeryTightSCF NoPop Conv

%casscf nel      5
        norb     5
        nroots   2
        mult     2
        end

%mrci  CIType      MRCI
       NewBlock 2 *
         excitations none
         NRoots 15
         refs CAS(5,5) end
         end
       NewBlock 4 *
         excitations none
         NRoots 15
         refs CAS(5,5) end
         end
       end

* xyz 0 2
C  0 0 0
H  0 0 1.15
*
```

This yields:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is    -38.308119994 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    2    -1    0    0      0.000      0.000       0.0
  1    2    -1    1    0      0.000      0.000       0.0
  2    4    -1    0    1     14.679      0.399    3221.6
  3    2    -1    2    0    126.464      3.441   27755.7
  4    2    -1    3    0    126.464      3.441   27755.7
  5    2    -1    4    0    132.689      3.611   29121.8
  6    2    -1    5    0    164.261      4.470   36051.2
  7    2    -1    6    0    305.087      8.302   66958.9
  8    2    -1    7    0    305.087      8.302   66958.9
  9    4    -1    1    1    328.911      8.950   72187.7
 10    4    -1    2    1    452.676     12.318   99350.8
 11    4    -1    3    1    452.676     12.318   99350.8
 12    2    -1    8    0    460.116     12.520  100983.9
 13    2    -1    9    0    463.438     12.611  101712.9
 14    2    -1   10    0    463.438     12.611  101712.9
 ...
```

Thus, if we want to focus on the low-lying states we should include five doublet and one quartet root. Now we run a second job with these roots and scan the internuclear distance.

```
! ano-pVDZ VeryTightSCF NoPop Conv MRCI+Q

%casscf nel     5
        norb    5
        nroots  5,1
        mult    2,4
        shiftup 2
        end

%paras R = 0.8,2.5,25
       end

* xyz 0 2
C  0 0 0
H  0 0 {R}
*
```

The surfaces obtained in this run are shown in Fig. 3.43. You can nicely see the crossing of the $^2\Sigma$ and $^2\Delta$ states fairly close to the equilibrium distance and also the merging of the $^4\Sigma$ state with $^2\Pi$ and $^2\Sigma$ towards the asymptote that where C-H dissociates in a neutral C-atom in its ^{3}P ground state and a neutral hydrogen atom in its ^{2}S ground state. You can observe that once `AllSingles` is set to true (the default), the default settings of the MRCI module yield fairly smooth potential energy surfaces.

Fig. 3.43: Potential energy surfaces for some low-lying states of CH using the MRCI+Q method

In many cases one will focus on the region around the minimum where the surface is nearly quadratic. In this case one

can still perform a few (2, 3, 5,) point polynomial fitting from which the important parameters can be determined. The numerical accuracy and the behavior with respect to T_{sel} has to be studied in these cases since the selection produces some noise in the procedure. We illustrate this with a calculation on the HF molecule:

```
! ano-pVDZ VeryTightSCF NoPop Conv MRCI+Q

%paras  R = 0.85,1.1,7
        end

%casscf nel     8
        norb    5
        nroots  1  mult    1
        shiftup 2.5 shiftdn 2.5 switchstep nr gtol    1e-5
        end

%mrci   tsel 1e-8
        tpre 1e-5
        end

* xyz 0 1
F  0 0 0
H  0 0 {R}
*
```

The output contains the result of a Morse fit:

```
Morse-Fit Results:
 Re               =   0.93014 Angstroem
 we               =   4111.2 cm**-1
 wexe             =     79.5 cm**-1
```

Which may be compared with the CCSD(T) values calculated with the same basis set:

```
Morse-Fit Results:
 Re               =   0.92246 Angstroem
 we               =   4209.8 cm**-1
 wexe             =     97.6 cm**-1
```

The agreement between MRCI+Q and CCSD(T) results is fairly good.

3.19.11 Multireference Systems - Ozone

The ozone molecule is a rather classical multireference system due to its diradical character. Let us look at the three highest occupied and lowest unoccupied MO (the next occupied MO is some 6 eV lower in energy and the next virtual MO some 10 eV higher in energy):

(a) (a) MO-9 (b) (b) MO-10 (c) (c) MO 11(HOMO)

(d) (d) MO 12(LUMO)

Fig. 3.44: Frontier MOs of the Ozone Molecule.

These MOs are two σ lone pairs which are high in energy and then the symmetric and antisymmetric combinations of the oxygen π lone pairs. In particular, the LUMO is low lying and will lead to strong correlation effects since the $(\text{HOMO})^2 \rightarrow (\text{LUMO})^2$ excitation will show up with a large coefficient. Physically speaking this is testimony of the large diradical character of this molecule which is roughly represented by the structure ↑O-O-O↓. Thus, the minimal active space to treat this molecule correctly is a CAS(2,2) space which includes the HOMO and the LUMO. We illustrate the calculation by looking at the RHF, MP2 MRACPF calculations of the two-dimensional potential energy surface along the O–O bond distance and the O-O-O angle (experimental values are 1.2717 Å and 116.78°).

```
! ano-pVDZ VeryTightSCF NoPop MRCI+Q Conv

%paras R     = 1.20,1.40,21
       Theta = 100,150,21
       end

%casscf nel     2
        norb    2
        mult    1
        nroots  1
        end

%mrci   tsel    1e-8
        tpre    1e-5
        end

* int 0 1
O  0 0 0  0   0         0
O  1 0 0 {R}  0         0
O  1 2 0 {R}  {Theta}   0
*
```

This is a slightly lengthy calculation due to the 441 energy evaluations required. RHF does not find any meaningful minimum within the range of examined geometries. MP2 is much better and comes close to the desired minimum but underestimates the O–O distance by some 0.03 Å. CCSD(T) gives a very good angle but a O–O distance that is too long. In fact, the largest doubles amplitude is ≈ 0.2 in these calculations (the HOMO–LUMO double excitation) which indicates a near degeneracy calculation that even CCSD(T) has problems to deal with. Already the

CAS(2,2) calculation is in qualitative agreement with experiment and the MRCI+Q calculation then gives almost perfect agreement.

The difference between the CCSD(T) and MRCI+Q surfaces shows that the CCSD(T) is a bit lower than the MRCI+Q one suggesting that it treats more correlation. However, CCSD(T) does it in an unbalanced way. The MRCI calculation employs single and double excitations on top of the HOMO-LUMO double excitation, which results in triples and quadruples that apparently play an important role in balancing the MR calculation. These excitations are treated to all orders explicitly in the MRCI calculation but only approximately (quadruples as simultaneous pair excitations and triples perturbatively) in the coupled-cluster approach. Thus, despite the considerable robustness of CC theory in electronically difficult situations it is not applicable to genuine multireference problems.

This is a nice result despite the too small basis set used and shows how important it can be to go to a multireference treatment with a physically reasonable active space (even if is only 2×2) in order to get qualitatively and quantitatively correct results.

(a) (a) RHF (b) (b) CASSCF(2,2) (c) (c) MP2 (d) (d) CCSD(T) (e) (e) MRCI+Q

(f) (f) Difference CCSD(T)/MRCI+Q

Fig. 3.45: 2D potential energy surface for the O_3 molecule calculated with different methods

3.19.12 Size Consistency

Finally, we want to study the size consistency errors of the methods. For this we study two non-interacting HF molecules at the single reference level and compare to the energy of a single HF molecule. This should give a reasonably fair idea of the typical performance of each method (energies in Eh)[5]:

```
E(HF)          E(HF+HF)   |Difference|
  CISD+Q   -100.138 475  -200.273 599     0.00335
  ACPF     -100.137 050  -200.274 010     0.00000
  ACPF2    -100.136 913  -200.273 823     0.00000
  AQCC     -100.135 059  -200.269 792     0.00032
```

The results are roughly as expected – CISD+Q has a relatively large error, ACPF and ACPF/2 are perfect for this type of example; AQCC is not expected to be size consistent and is (only) about a factor of 10 better than CISD+Q in this respect. CEPA-0 is also size consistent.

[5] Most of these numbers were obtained with a slightly older version but will not change too much in the present version.

3.19.13 Efficient MR-MP2 Calculations for Larger Molecules

Uncontracted MR-MP2 approaches are nowadays outdated. They are much more expensive than internally contracted e.g. the NEVPT2 method described in section *N-Electron Valence State Perturbation Theory (NEVPT2)*. Moreover, MR-MP2 is prone to intruder states, which is a major obstacle for practical applications. For historical reasons, this section is dedicated to the traditional MR-MP2 approach that is available since version 2.7.0 ORCA. The implementation avoids the full integral transformation for MR-MP2 which leads to significant savings in terms of time and memory. Thus, relatively large RI-MR-MP2 calculations can be done with fairly high efficiency. However, the program still uses an uncontracted first order wavefunction which means that for very large reference space, the calculations still become untractable.

Consider for example the rotation of the stilbene molecule around the central double bond

(a) (a) (b) (b)

Fig. 3.46: Rotation of stilbene around the central double bond using a CASSCF(2,2) reference and correlating the reference with MR-MP2.

The input for this calculation is shown below. The calculation has more than 500 basis functions and still runs through in less than one hour per step (CASSCF-MR-MP2). The program takes care of the reduced number of two-electron integrals relative to the parent MRCI method and hence can be applied to larger molecules as well. Note that we have taken a "JK" fitting basis in order to fit the Coulomb and the dynamic correlation contributions both with sufficient accuracy. Thus, this example demonstrates that MR-MP2 calculations for not too large reference spaces can be done efficiently with ORCA (as a minor detail note that the calculations were started at a dihedral angle of 90 degrees in order to make sure that the correct two orbitals are in the active space, namely the central carbon p-orbitals that would make up the pi-bond in the coplanar structure).

```
#
# Stilbene rotation using MRMP2
#
! def2-TZVP def2/JK RIJCOSX RI-MRMP2

%casscf nel            2
        norb           2
        end

%mrci   maxmemint 2000
        tsel 1e-8
        end


%paras  DIHED = 90,270, 19
        end

* int 0 1
```

(continues on next page)

(continued from previous page)

```
 C     0   0   0   0.000000     0.000     0.000
 C     1   0   0   1.343827     0.000     0.000
 C     2   1   0   1.490606   125.126     0.000
 C     1   2   3   1.489535   125.829   \{DIHED\}
 C     4   1   2   1.400473   118.696   180.000
 C     4   1   2   1.400488   122.999     0.000
 C     6   4   1   1.395945   120.752   180.000
 C     5   4   1   1.394580   121.061   180.000
 C     8   5   4   1.392286   120.004     0.000
 C     3   2   1   1.400587   118.959   180.000
 C     3   2   1   1.401106   122.779     0.000
 C    11   3   2   1.395422   120.840   180.001
 C    12  11   3   1.392546   120.181     0.000
 C    13  12  11   1.392464   119.663     0.000
 H     1   2   3   1.099419   118.266     0.000
 H     2   1   3   1.100264   118.477   179.999
 H     5   4   1   1.102119   119.965     0.000
 H     6   4   1   1.100393   121.065     0.000
 H     7   6   4   1.102835   119.956   180.000
 H     8   5   4   1.102774   119.989   180.000
 H     9   8   5   1.102847   120.145   180.000
 H    10   3   2   1.102271   120.003     0.000
 H    11   3   2   1.100185   121.130     0.000
 H    12  11   3   1.103001   119.889   180.000
 H    13  12  11   1.102704   120.113   180.000
 H    14  13  12   1.102746   119.941   180.000
*
```

3.19.14 Properties Calculation Using the SOC Submodule

Zero-Field Splitting

The spin-orbit coupling (SOC) and spin-spin coupling (SSC) contributions to the zero-field splitting (ZFS) can be calculated very accurately using a wavefunction obtained from a multiconfigurational calculation of a multi-reference type such as CASSCF, MRCI, or MRPT as is described in QDPT Magnetic Properties Section *Magnetic Properties Through Quasi Degenerate Perturbation Theory*.

```
# In case that you want to run QDPT-SOC calculation with manually adjusted␣
↪diagonal
# energies you can copy the following part into the %mrci soc block and modify it␣
↪as needed
# (energies are given in wavenumbers relative to the lowest state).              ␣
↪
#                                                                                ␣
↪
# NOTE: It is YOUR responsibility to make sure that the CAS-CI state that you may␣
↪want to dress
#       with these energies correlates properly with the energies printed here.␣
↪The order of states
#       or even the identity of states may change with and without inclusion of␣
↪dynamic correlation.
#       In the case that dynamic correlation strongly mixes, different CAS-CI␣
↪states there may not
#        even be a proper correlation!                                           ␣
↪
#                                                                                ␣
↪
     EDiag[  0]    -149.526236244  # root   0 of block 0
     EDiag[  1]    -149.359818263  # root   1 of block 0
```

(continues on next page)

(continued from previous page)

```
     EDiag[  2]    -149.359818263  # root   2 of block 0
     EDiag[  3]    -149.496737695  # root   0 of block 1
     EDiag[  4]    -149.496737695  # root   1 of block 1
     EDiag[  5]    -149.474844108  # root   2 of block 1
```

Those transition energies can be substituted by a more accurate energies provided in the input file as follows:

```
%soc
dosoc true
dossc true
     EDiag[  0]    -149.526236244  # root   0 of block 0
     EDiag[  1]    -149.359818263  # root   1 of block 0
     EDiag[  2]    -149.359818263  # root   2 of block 0
     EDiag[  3]    -149.496737695  # root   0 of block 1
     EDiag[  4]    -149.496737695  # root   1 of block 1
     EDiag[  5]    -149.474844108  # root   2 of block 1
end
```

Accurate diagonal energies generally improve the accuracy of the SOC and SSC splittings.

Local Zero-Field Splitting

The submodule can also be used to calculate the local ZFS splitting parameters of atomic centers. The method, referred to as local complete active space configuration interaction (L-CASCI), can be used to separate into atomic contributions the SOC part of the total ZFS tensor. The rational behind it and additional details are described in the original publication [509]; below are listed only the steps required to reproduce the calculation for the dimer complex presented there.

1. The first step consists in obtaining the molecular orbitals that are going to be used in the configuration interaction (CI) procedure. A good set of orbitals can be obtained from a restricted open-shell spin-averaged Hartree-Fock (SAHF) calculation. The relevant part of the input is listed below:

```
! def2-TZVP

% scf
    hftyp rohf
    rohf_case sahf
    rohf_numop 2
    rohf_nel[1] 9
    rohf_norb[1] 10
    end
```

For the present Mn(II)Mn(III) dimer there are a total of 9 electrons distributed into 10 d-orbitals.

2. Next, the molecular orbitals are localized using one of the implemented localization schemes. Below is the `orca_loc` input used in this case:

```
sahf.gbw
sahf.loc
0
200 # first of the 10 d-orbitals
209 # last of the 10 d-orbitals
128
0.000001
0.75
0.65
2
```

3. Following this, the localized orbitals are made locally canonical by block diagonalizing the Fock matrix using the `orca_blockf` utility.

```
orca_blockf sahf.gbw sahf.loc 200 204 205 209
```

The first two numbers define the range of molecular orbitals localized on one center; the last two are for the second center.

4. The recanonicalized orbitals stored in the `sahf.loc` file can be then used to calculate the SOC contribution to the local ZFS of the Mn(III) center using the following MRCI input:

```
! ZORA-def2-TZVP def2-TZVP/C ZORA
! nomulliken noloewdin
! moread noiter allowrhf
! moread

% mrci
    citype mrci
    tsel 0
    tpre 0
    intmode ritrafo
    solver diis
    soc
        intmode ritrafo
        dosoc true
        end
    newblock 10 *
        nroots 5
        excitations none
        refs
            #  Mn(II)    Mn(III)
            {1 1 1 1 1  1 1 1 1 0}
            {1 1 1 1 1  1 1 1 0 1}
            {1 1 1 1 1  1 1 0 1 1}
            {1 1 1 1 1  1 0 1 1 1}
            {1 1 1 1 1  0 1 1 1 1}
            end
        end
    newblock 8 *
        nroots 45
        excitations none
        refs
            #  Mn(II)    Mn(III)
            {1 1 1 1 1  2 1 1 0 0}
            {1 1 1 1 1  2 1 0 1 0}
            {1 1 1 1 1  2 1 0 0 1}
            {1 1 1 1 1  2 0 1 1 0}
            {1 1 1 1 1  2 0 1 0 1}
            {1 1 1 1 1  2 0 0 1 1}
            {1 1 1 1 1  1 2 1 0 0}
            {1 1 1 1 1  1 2 0 1 0}
            {1 1 1 1 1  1 2 0 0 1}
            {1 1 1 1 1  1 1 2 0 0}
            {1 1 1 1 1  1 1 1 1 0}
            {1 1 1 1 1  1 1 1 0 1}
            {1 1 1 1 1  1 1 0 2 0}
            {1 1 1 1 1  1 1 0 1 1}
            {1 1 1 1 1  1 1 0 0 2}
            {1 1 1 1 1  1 0 2 1 0}
            {1 1 1 1 1  1 0 2 0 1}
            {1 1 1 1 1  1 0 1 2 0}
            {1 1 1 1 1  1 0 1 1 1}
            {1 1 1 1 1  1 0 1 0 2}
            {1 1 1 1 1  1 0 0 2 1}
            {1 1 1 1 1  1 0 0 1 2}
```

(continues on next page)

(continued from previous page)

```
            {1 1 1 1 1  0 2 1 1 0}
            {1 1 1 1 1  0 2 1 0 1}
            {1 1 1 1 1  0 2 0 1 1}
            {1 1 1 1 1  0 1 2 1 0}
            {1 1 1 1 1  0 1 2 0 1}
            {1 1 1 1 1  0 1 1 2 0}
            {1 1 1 1 1  0 1 1 1 1}
            {1 1 1 1 1  0 1 1 0 2}
            {1 1 1 1 1  0 1 0 2 1}
            {1 1 1 1 1  0 1 0 1 2}
            {1 1 1 1 1  0 0 2 1 1}
            {1 1 1 1 1  0 0 1 2 1}
            {1 1 1 1 1  0 0 1 1 2}
            end
        end
    end
```

5. The three second order ZFS components printed at the end of the calculation (`Second order D-tensor: component 0`, etc.) are scaled using the *S* value for the complex, which in this case is 4.5 (9 electrons $\times$ 0.5). In order to obtain the correct local value of the ZFS, the three matrices have to be rescaled using the *S* value for Mn(III), which is to 2. Note that the three matrices have different scaling prefactors, and the dependence on *S* is not the same:

$\mathbf{D}^{SOC-(0)} \propto \frac{1}{S^2}$

$\mathbf{D}^{SOC-(-1)} \propto \frac{1}{S(2S-1)}$

$\mathbf{D}^{SOC-(+1)} \propto \frac{1}{(S+1)(2S+1)}$

These equations can be used to calculate the required prefactors. For example in the case of the *SOC*-(0) the prefactor is equal to:

$\mathbf{D}^{SOC-(0)}_{\text{Mn(III)}} = \frac{4.5^2}{2^2} \cdot \mathbf{D}^{SOC-(0)}_{\text{dimer}} = 5.0625 \cdot \mathbf{D}^{SOC-(0)}_{\text{dimer}}$

The final step is to scale the two remaining matrices using the appropriate prefactors, sum all three of them up, diagonalize the resulting the matrix, and use its eigenvalues to calculate the *D* and *E* parameters. These represent the local ZFS parameters of the Mn(III) center.

Zero-Field Splitting from an excited Multiplet

For an excited state Multiplet the Calculationof ZFS can be requested by

```
Lowest eigenvalue of the SOC matrix:    -149.86223277 Eh
Energy stabilization:    -2.54512 cm-1
Eigenvalues:     cm-1         eV      Boltzmann populations at T =  300.000 K
   0:            0.00        0.0000       3.36e-01
   1:            2.37        0.0003       3.32e-01
   2:            2.37        0.0003       3.32e-01
   3:         7757.65        0.9618       2.33e-17
   4:         7757.66        0.9618       2.33e-17
   5:        11913.81        1.4771       5.15e-26
```

```
soc
 DTensor true
 IStates 3,4,5
end
```

```
****************************************
 EXCITED STATE ZERO-FIELD SPLITTING:
****************************************
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------
Computing Excited State D Tensors of
Excited State Multiplet Consisting of States :  3  4  5
-------------------------------------------
          0         4
          1         5
          2         0
          3         1
          4         0
          5         2


-------------------------------------------
           ZERO-FIELD SPLITTING
(2ND ORDER SPIN-ORBIT COUPLING CONTRIBUTION)
-------------------------------------------

D   =   -2.668445  cm-1
E/D =    0.000103


...

-------------------------------------------------------
                  ZERO-FIELD SPLITTING
EFFECTIVE HAMILTONIAN SOC CONTRIBUTION
-------------------------------------------------------

D   =   -2.674495  cm-1
E/D =    0.009610

...
```

g-Tensor

The `orca_mrci` program contains an option to calculate g-tensors using MRCI wavefunctions. For a system with an odd number of electrons, the doubly degenerate eigenvalues obtained from the QDPT procedure represent Kramers pairs, which are used to build the matrix elements of the total spin operator and the total angular momentum operator from the Zeeman Hamiltonian. Denoting Ψ as a solution and $\bar{\Psi}$ as its Kramers partner and using matrix element notations

$$\Phi_{11}^k = \langle\Psi|\,\hat{L}_k + g_e\hat{S}_k\,|\Psi\rangle\,,\ \Phi_{12}^k = \langle\Psi|\,\hat{L}_k + g_e\hat{S}_k\,|\bar{\Psi}\rangle\,,\ k = x, y, z \tag{3.133}$$

The elements of g-matrix are obtained as:

$$g_{kz} = 2\Phi_{11}^k,\ g_{ky} = -2\Im\left(\Phi_{12}^k\right),\ g_{kx} = 2\Re\left(\Phi_{12}^k\right) \tag{3.134}$$

Then, the true tensor G is built from g-matrices:

$$G = gg^T \tag{3.135}$$

G is subjected further to diagonalization yielding positive eigenvalues, the square roots of which give the principal values of g-matrix.

$$g_{xx} = \sqrt{G_{xx}},\ g_{yy} = \sqrt{G_{yy}},\ g_{zz} = \sqrt{G_{zz}} \tag{3.136}$$

A typical mrci block of the input file for a g-tensor calculation should (e.g. for a S=3/2 problem) look as the following:

```
%mrci   ewin -4,1000
        citype mrci
        cimode direct2
        intmode fulltrafo
        solver diis
        etol 1e-8
        rtol 1e-8
        tsel 1e-6
        tpre 1e-5
        soc
          PrintLevel 2
          GTensor true      # make g-tensor calculations
          NDoubGTensor 2    # number of Kramers doublets to account
                            # for every pair a separate
                            # calculation is performed
        end
        newblock 4 *
          excitations cisd
          nroots 10
          refs  cas(7,5) end
        end
end
```

The result for the first Kramers pair is printed as follows:

```
--------------
KRAMERS PAIR 1
--------------

Matrix elements Re<1|S|1>   -0.072128   0.024511  -2.998843
Matrix elements Re<1|S|2>   -0.001088   0.000366  -0.002010
Matrix elements Im<1|S|2>   -0.000354  -0.001037  -0.000173
Matrix elements Re<1|L|1>   -0.027067   0.009209  -1.123531
Matrix elements Re<1|L|2>   -0.000031   0.000010  -0.000763
Matrix elements Im<1|L|2>   -0.000006  -0.000011  -0.000065

-------------------
ELECTRONIC G-MATRIX
-------------------

g-matrix:
-0.002240     0.000754    -0.005551
0.000720     0.002100     0.000477
-0.198556     0.067498    -8.251703

g-factors:
0.002220     0.002222     8.254370 iso =     2.752937

g-shifts:
-2.000100    -2.000098     6.252051 iso =     0.750618

Eigenvectors:
0.057426     0.998060     0.024055
0.998327    -0.057244    -0.008177
0.006784    -0.024484     0.999677
```

Here for the L and S matrix elements indices 1 and 2 are assumed to denote Kramers partners, and three numbers in the first row stand for x, y, z contributions.

In addition the g-tensor is calculated within the Effective Hamiltonian formalism.

```
---------------------------------------------
```

(continues on next page)

(continued from previous page)

```
ELECTRONIC G-MATRIX FROM EFFECTIVE HAMILTONIAN
----------------------------------------------

g-matrix:
1.978874    -0.000345     0.018908
-0.000345     1.977899    -0.006433
0.018879    -0.006418     2.763402

g-factors:
1.977789     1.978477     2.763909 iso =     2.240058

g-shifts:
-0.024530    -0.023843     0.761590 iso =     0.237739

Eigenvectors:
0.288884     0.957062     0.024060
0.957364    -0.288770    -0.008181
0.000882    -0.025397     0.999677

# The g-factors are square roots of the eigenvalues of gT*g
# Orientations are the eigenvectors of gT*g
```

Finally and only within the MRCI module the g-tensor is evaluated by using the Sum Over States formalism[510]:

```
-----------------------------------------------------------------------------
SUM OVER STATES CALCULATION OF THE SPIN HAMILTONIAN (for g and HFC tensors)
-----------------------------------------------------------------------------


Ground state index        =     0
Ground state multiplicity =     4
Ground state spin density = P[  1]
State =   1  <0|P|I>=   2  <0|Q|I>=  19
State =   2  <0|P|I>=   3  <0|Q|I>=  27
State =   3  <0|P|I>=   4  <0|Q|I>=  34
State =   4  <0|P|I>=   5  <0|Q|I>=  40
State =   5  <0|P|I>=   6  <0|Q|I>=  45
State =   6  <0|P|I>=   7  <0|Q|I>=  49
State =   7  <0|P|I>=   8  <0|Q|I>=  52
State =   8  <0|P|I>=   9  <0|Q|I>=  54
State =   9  <0|P|I>=  10  <0|Q|I>=  55

Origin for angular momentum        ... ( -0.0006, -0.0010,  0.0021)
Kinetic Energy                     ... done
Relativistic mass correction       ... done
Gauge correction                   ... done
Angular momentum integrals         ... done
Reading Spin-Orbit Integrals       ... done
-----------------------
MATRIX ELEMENT PRINTING
-----------------------

Energy differences (DE=EI-E0) and spin-orbit matrix elements (SO=<I|HSO|0>) are
printed in cm**-1. Orbital Zeeman matrix elements (L=<I|L|0>) are printed in au.

State     DE         LX        LY        LZ           SOX         SOY         SOZ

1    1349.3     0.0464   -0.0158    1.9264     -23.432       7.965    -974.312
2   13026.2    -0.6596    0.6888    0.0214     337.028    -351.116     -10.966
3   13615.1    -0.6961   -0.6514    0.0113     354.225     332.219      -5.736
4   56686.3    -0.0053    0.0077    0.0971       1.794      -1.696     -36.786
```

(continues on next page)

(continued from previous page)

```
5   56954.2    -0.0516   -0.0048   -0.0042      28.211       5.821       1.459
6   56994.0    -0.0418    0.0233   -0.0025      15.185      -2.144       1.145
7   63371.5    -0.0211    0.0226    0.0078       3.833      -2.948      -2.724
8   64176.0    -0.0652    0.0032   -0.0002      32.779       6.146       0.063
9   74309.9    -0.0007    0.0032    0.0380       0.183      -1.058     -13.517

-------------------
ELECTRONIC G-MATRIX
-------------------

raw-matrix :
2.025533    -0.000738     0.021755
-0.000738     2.024537    -0.007389
0.021755    -0.007389     2.928943

g-factors:
2.024122     2.025363     2.929527 iso =     2.326338

g-shifts:
0.021803     0.023044     0.927208 iso =     0.324018

Eigenvectors:
0.533896    -0.845208     0.024064
0.845530     0.533866    -0.008182
-0.005932     0.024715     0.999677


Euler angles w.r.t. molecular frame (degrees):
-76.5038       1.4564    -161.2223

-----------------------------
CONTRIBUTIONS TO THE G-MATRIX
-----------------------------

Term                                 g1           g2           g3
-----------------------------------------------------------------
Relativistic mass correction:   -0.0008220   -0.0008220   -0.0008220
Gauge correction            :    0.0000000    0.0000000    0.0000000
g(OZ/SOC)                   :    0.0226250    0.0238662    0.9280297

State   1                               :    0.0000000   -0.0000000    0.9279829
State   2                               :    0.0013767    0.0223913    0.0000000
State   3                               :    0.0212332    0.0014408    0.0000000
State   4                               :    0.0000000    0.0000004    0.0000418
State   5                               :    0.0000074    0.0000099    0.0000001
State   6                               :    0.0000002    0.0000078    0.0000001
State   7                               :    0.0000000    0.0000015    0.0000002
State   8                               :    0.0000076    0.0000144    0.0000000
State   9                               :    0.0000000    0.0000000    0.0000046
----------------------------------------
Total g-shifts              :    0.0218030    0.0230442    0.9272077


# The g-factors are square roots of the eigenvalues of gT*g
# Orientations are the eigenvectors of gT*g
```

Note that within the SOS formalism in addition to the second order (SOC) contributions the bilinear to the field terms: Relativistic mass correction and diamagnetic spin-orbit term (Gauge) are evaluated. As can be seen these corrections are rather negligible in comparison to the second order SOC contributions and most of the time can be safely omitted. Moreover further insight is obtained by printing the individual contribution of each excited state to the g-tensor. In the example above the first excited state contributes to the g_z component while the next two to both the g_x and g_y

components, respectively.

So to summarize the g-tensor calculations in the framework of wavefunction based methods like MRCI and/or CASSCF can be evaluated:

- via the QDPT approach within an individual Kramers doublet. This is valid analysis only for non-integer spin cases. In particular for systems with well isolated Kramers doublets where the EPR spectrum originates only from one Kramers doublet defined within the pseudo spin 1/2 formalism. This analysis has been proven useful in determining the sign of the ZFS and the electronic structure of the system under investigation.[511]
- within the effective Hamiltonian approach. This is a valid analysis for all spin cases as it provides the principal g-values of the system under investigation evaluated in the molecular axis frame. These g-values can be directly compared with the experimentally determined ones.[512]
- within the sum over states formalism (SOS). As above this analysis is valid for all spin cases and is only available via the MRCI module.

Magnetization and Magnetic Susceptibility

The MRCI and CASSCF modules of ORCA allow for the calculation of magnetization and magnetic susceptibility curves at different fields and temperatures by differentiation of the QDPT Hamiltonian with respect to the magnetic field. For magnetic susceptibility, calculations are performed in two ways when a static field different from zero is defined: (i) as the second derivative of energy with respect to the magnetic field and (ii) as the magnetization divided by the magnetic field. Although the first method corresponds to the definition of magnetic susceptibility, the second approach is widely used in the experimental determination of $\chi * T$ curves. If the static field is low, both formulas tend to provide similar values.

The full list of keywords is presented below.

```
%mrci
      citype mrci
      newblock 3 *
      excitations none
      refs cas(2,7) end
      end
      soc
        dosoc true
        domagnetization true       # Calculate magnetization (def: false)
        dosusceptibility true      # Calculate susceptiblity (def: false)
        LebedevPrec            5   # Precision of the grid for different field
                                   # directions (meaningful values range from 1
                                   # (smallest) to 10 (largest))
        nPointsFStep           5   # number of steps for numerical differentiation
                                   # (def: 5, meaningful values are 3, 5 7 and 9)
        MAGFieldStep       100.0   # Size of field step for numerical␣
↪differentiation
                                   # (def: 100 Gauss)
        MAGTemperatureMIN    4.0   # minimum temperature (K) for magnetization
        MAGTemperatureMAX    4.0   # maximum temperature (K) for magnetization
        MAGTemperatureNPoints  1   # number of temperature points for magnetization
        MAGFieldMIN          0.0   # minimum field (Gauss) for magnetization
        MAGFieldMAX      70000.0   # maximum field (Gauss) for magnetization
        MAGNpoints            15   # number of field points for magnetization
        SUSTempMIN           1.0   # minimum temperature (K) for susceptibility
        SUSTempMAX         300.0   # maximum temperature (K) for susceptibility
        SUSNPoints           300   # number of temperature points for␣
↪susceptibility
        SUSStatFieldMIN      0.0   # minimum static field (Gauss) for␣
↪susceptibility
        SUSStatFieldMAX      0.0   # maximum static field (Gauss) for␣
↪susceptibility
        SUSStatFieldNPoints    1   # number of static fields for susceptibility
```

(continues on next page)

(continued from previous page)

```
     end
end
```

The same keywords apply for CASSCF calculations in rel block (instead of soc in MRCI). Although different aspects of integration and grid precision can be modified through keywords, default values should provide an accurate description of both properties. Calculated magnetization and susceptibility are printed in .sus and .mag files, respectively and also in the output file.

```
-------------------------------------------------------------------------------
FIELD DEPENDENT MAGNETIZATION AND MEAN SUSCEPTIBILITY (chi=M/B)
-------------------------------------------------------------------------------
TEMPERATURE (K)   M. FIELD (Gauss)   MAGNETIZATION (B.M.)   chi*T (cm3*K/mol)
-------------------------------------------------------------------------------

4.00               0.00               0.000000                  inf
4.00            5000.00               0.350759             1.567189
4.00           10000.00               0.688804             1.538788
4.00           15000.00               1.003466             1.494496
4.00           20000.00               1.287480             1.438115
4.00           25000.00               1.537346             1.373773
4.00           30000.00               1.752841             1.305282
4.00           35000.00               1.936067             1.235764
4.00           40000.00               2.090450             1.167516
4.00           45000.00               2.219920             1.102067
4.00           50000.00               2.328368             1.040315
4.00           55000.00               2.419335             0.982690
4.00           60000.00               2.495883             0.929301
4.00           65000.00               2.560582             0.880052
4.00           70000.00               2.615538             0.834730
-------------------------------------------------------------
```

```
-------------------------------------------------------------
  TEMPERATURE DEPENDENT MAGNETIC SUSCEPTIBILITY
  -------------------------------------------------------------
  STATIC  FIELD    TEMPERATURE       chi*T (cm3*K/mol)
  (Gauss)            (K)          M/B          d2E/dB2
  -------------------------------------------------------------

  0.00       1.00           ----       1.576836
  0.00       2.00           ----       1.576910
  0.00       3.00           ----       1.576951
  0.00       4.00           ----       1.576988
  0.00       5.00           ----       1.577023
  0.00       6.00           ----       1.577057
  0.00       7.00           ----       1.577091
  0.00       8.00           ----       1.577125
  0.00       9.00           ----       1.577159
  0.00      10.00           ----       1.577193
  0.00      11.00           ----       1.577227
  .....
  0.00     300.00           ----       1.586942
  1000.00       1.00       1.570517       1.558042
  1000.00       2.00       1.575324       1.572178
  1000.00       3.00       1.576246       1.574845
  1000.00       4.00       1.576590       1.575802
  1000.00       5.00       1.576768       1.576264
  1000.00       6.00       1.576880       1.576530
  1000.00       7.00       1.576961       1.576704
  1000.00       8.00       1.577026       1.576829
  .....
```

Note that the CASSCF module also supports the calculation of susceptibility tensors at non-zero user-defined magnetic fields. This is not yet possible with the MRCI module.

MCD and Absorption Spectra

The MRCI module of the ORCA program allows calculating MCD spectra and the SOC effects on absorption spectra. The formalism is described in detail by Ganyushin and Neese[513]. The approach is based on the direct calculation of the transition energies and transition probabilities between the magnetic levels. Namely, the differential absorption of LCP- and RCP photons for transitions from a manifold of initial states A to a manifold of final states J. Using Fermi's golden rule, the Franck-Condon approximation, assuming a pure electronic dipole mechanism and accounting for the Boltzmann populations of the energy levels, the basic equation of MCD spectroscopy may be written as (atomic units are used throughout):

$$\frac{\Delta\varepsilon}{E} = \gamma \sum_{a,j} \left(N_a - N_j\right) \left(\left|\langle \Psi_a \left| m_{\text{LCP}} \right| \Psi_j \rangle\right|^2 - \left|\langle \Psi_a \left| m_{\text{RCP}} \right| \Psi_j \rangle\right|^2\right) f\left(E\right) \tag{3.137}$$

Here a and j label members of the initial and state manifold probed in the experiments.

$$N_a\left(B,T\right) = \frac{\exp\left(-E_a/kT\right)}{\sum\limits_i \exp\left(-E_i/kT\right)} \tag{3.138}$$

denotes the Boltzmann population and if the a-th ground state sublevel at energy E_a, $f\left(E\right)$ stands for a line shape function, and γ denotes a collection of constants. The electric dipole operators are given by:

$$m_{\text{LCP}} \equiv m_x - i m_y \tag{3.139}$$

$$m_{\text{RCP}} \equiv m_x + i m_y \tag{3.140}$$

They represent linear combinations of the dipole moment operator:

$$\vec{m} = \sum_N Z_N \vec{R}_N - \sum_i \vec{r}_i \tag{3.141}$$

where N and i denotes summations of nuclei (at positions $\vec{R}_N$ with charges Z_N) and electrons (at positions $\vec{r}_i$) respectively. The calculated transition dipole moment are subjected to the space averaging over the Euler angles which is performed by a simple summation over three angular grids.

$$\left(\frac{\Delta\varepsilon}{E}\right)_{ev} = \frac{1}{8\pi^2} \int\limits_{\psi=0}^{2\pi} \int\limits_{\phi=0}^{2\pi} \int\limits_{\theta=0}^{\pi} \left(\frac{\Delta\varepsilon}{E}\right) \sin\theta d\theta d\phi d\psi \approx \sum_{\mu\eta\tau} \left(\frac{\Delta\varepsilon}{E}\right)_{\mu\eta\tau} \sin\theta_\tau \tag{3.142}$$

Finally, every transition is approximated by a Gaussian curve with a definite Gaussian shape width parameter. Hence, the final calculated MCD spectrum arises from the superposition of these curves.

As an illustration, consider calculation of a classical example of MCD spectrum of [Fe(CN) $_6$]$^{3-}$. The mrci block of the input file is presented below.

```
%mrci   ewin -4,10000
        citype mrddci2
        intmode ritrafo
        Tsel  1e-6
        Tpre  1e-5
        etol 1e-8
        rtol 1e-8
        cimode direct2
        maxmemint 300
        solver diis
        davidsonopt 0
        nguessmat 150
        MaxIter 50
```

(continues on next page)

(continued from previous page)

```
        LevelShift 0.5
        PrintLevel 3
        soc
        printlevel 3
        Domcd true        # perform the MCD calculation
        NInitStates 24    # number of SOC and SSC state to account
                          # Starts from the lowest state
        NPointsTheta 10   # number of integration point for
        NPointsPhi 10     # Euler angles
        NPointsPsi 10     #
        B 43500           # experimental magnetic field strength
                          # in Gauss
        Temperature 299.0 # experimental temperature (in K)
        end
        newblock 2 *
         nroots 12
         excitations cisd
         refs cas(23,12) end
         end
        end
```

The parameters B and Temperature can be assigned in pairs, i.e. B = 1000, 2000, 3000…, Temperature = 4, 10, 300…. The program calculates the MCD and absorption spectra for every pair. Now for every point of the integration grid the program prints out the Euler angles, the orientation of the magnetic field in the coordinate system of a molecule, and the energy levels.

```
Psi = 36.000 Phi = 72.000 Theta = 20.000

Bx = 8745.0 By = 12036.5 Bz = 40876.6

Energy levels (cm-1,eV):Boltzmann populations for T = 299.000   K
   0 :         0.000     0.0000         4.53e-01
   1 :         3.943     0.0005         4.45e-01
   2 :       454.228     0.0563         5.09e-02
   3 :       454.745     0.0564         5.08e-02
   4 :      1592.142     0.1974         2.13e-04
   5 :      1595.272     0.1978         2.10e-04
   6 :     25956.363     3.2182         2.59e-55
   7 :     25958.427     3.2184         2.56e-55
   8 :     25985.656     3.2218         2.25e-55
   9 :     25987.277     3.2220         2.23e-55
  10 :     26070.268     3.2323         1.49e-55
  11 :     26071.484     3.2325         1.49e-55
  12 :     31976.645     3.9646         6.78e-68
  13 :     31979.948     3.9650         6.67e-68
  14 :     32018.008     3.9697         5.56e-68
  15 :     32021.074     3.9701         5.48e-68
  16 :     32153.427     3.9865         2.90e-68
  17 :     32157.233     3.9870         2.84e-68
  18 :     42299.325     5.2444         1.81e-89
  19 :     42303.461     5.2450         1.78e-89
  20 :     42346.521     5.2503         1.45e-89
  21 :     42348.023     5.2505         1.44e-89
  22 :     42456.119     5.2639         8.53e-90
  23 :     42456.642     5.2640         8.51e-90
```

In the next lines, ORCA calculates the strength of LCP and RCP transitions and prints the transition energies, the difference between LCP and RCP transitions (denoted as C), and sum of LCP and RCP transitions (denoted as D), and C by D ratio.

```
dE     Na          C            D         C/D

  0 -> 1        3.943 4.53e-01  1.14e-13  8.13e-13   0.00e+00
  0 -> 2      454.228 4.53e-01  5.01e-09  9.90e-09   5.06e-01
  0 -> 3      454.745 4.53e-01 -4.65e-09  7.00e-09  -6.65e-01
  0 -> 4     1592.142 4.53e-01 -8.80e-08  1.02e-07  -8.67e-01
  0 -> 5     1595.272 4.53e-01 -2.29e-08  2.97e-08  -7.71e-01
  0 -> 6    25956.363 4.53e-01  1.22e+01  9.60e+01   1.27e-01
  0 -> 7    25958.427 4.53e-01  3.44e+01  3.52e+01   9.77e-01
  0 -> 8    25985.656 4.53e-01  3.83e+01  1.70e+02   2.25e-01
  0 -> 9    25987.277 4.53e-01 -7.73e+00  6.03e+01  -1.28e-01
  0 ->10    26070.268 4.53e-01 -6.11e+00  2.85e+01  -2.14e-01
  0 ->11    26071.484 4.53e-01  6.17e+00  9.21e+00   6.70e-01
  0 ->12    31976.645 4.53e-01  2.45e+01  6.21e+01   3.95e-01
  0 ->13    31979.948 4.53e-01 -6.58e+01  6.93e+01  -9.50e-01
  0 ->14    32018.008 4.53e-01  3.42e-01  1.07e+02   3.21e-03
  0 ->15    32021.074 4.53e-01 -6.16e+00  3.24e+01  -1.90e-01
  0 ->16    32153.427 4.53e-01 -4.73e+01  1.37e+02  -3.46e-01
  0 ->17    32157.233 4.53e-01 -1.02e+00  5.97e+01  -1.71e-02
  0 ->18    42299.325 4.53e-01  6.47e+00  2.11e+01   3.07e-01
  0 ->19    42303.461 4.53e-01 -2.59e+00  7.61e+00  -3.40e-01
  0 ->20    42346.521 4.53e-01  1.90e+01  8.99e+01   2.11e-01
  0 ->21    42348.023 4.53e-01  3.36e+00  3.55e+00   9.48e-01
  0 ->22    42456.119 4.53e-01  2.52e-01  4.86e-01   5.20e-01
  0 ->23    42456.642 4.53e-01 -2.01e+00  2.91e+00  -6.91e-01
  1 -> 2      450.285 4.45e-01  4.59e-09  6.87e-09   6.69e-01
  1 -> 3      450.802 4.45e-01 -4.96e-09  9.73e-09  -5.09e-01
```

All C and D values are copied additionally into the text files input.1.mcd, input.2.mcd…, for every pair of Temperature and B parameters. These files contain the energies and C and D values for every calculated transition. These files are used by the program `orca_mapspc` to calculate the spectra lines. The `orca_mapspc` program generates from the raw transitions data into spectra lines. The main parameters of the `orca_mapspc` program are described in section 7.18.1. A typical usage of the `orca_mapspc` program for MCD spectra calculation for the current example may look as the following:

```
orca_mapspc input.1.mcd MCD -x020000 -x150000 -w2000
```

Here the interval for the spectra generation is set from 20000 cm^{-1} to 50000 cm^{-1}, and the line shape parameter is set to 2000 cm^{-1}.

Very often, it is desirable to assign different line width parameters to different peaks of the spectra to obtain a better fitting to experiment. `orca_mapspc` can read the line shape parameters from a simple text file named as input.1.mcd.inp. This file should contain the energy intervals (in cm^{-1}) and the line shape parameters for this energy interval in the form of:

```
20000 35000 1000
35000 40000 2000
40000 50000 1000
```

This file should not be specified in the executing command; `orca_mapspc` checks for its presence automatically:

```
orca_mapspc input.1.mcd MCD -x020000 -x150000
Mode is MCD
Number of peaks            ... 276001
Start wavenumber [cm-1]    ...  20000.0
Stop wavenumber [cm-1]     ...  50000.0
Line width parameters are taken from the file:input.1.mcd.inp
Number of points           ... 1024
```

Finally, the `orca_mapspc` program generates the output text file input.1.mcd.dat which contains seven columns of numbers: transition energies, intensities of MCD transitions (the MCD spectrum), intensities of absorption transitions

(the absorption spectrum), the ratio between the MCD and absorption intensities, and the last three columns represent the "sticks" of the corresponding transitions.

```
Energy      C         D         C/D       C        D       E/D
24310.8   0.6673    980.2678   0.0006   0.0000   0.0000   0.0000
24340.1   0.8471   1174.3637   0.0007  -0.0001   0.0129  -0.0112
24369.5   1.0664   1408.5788   0.0007   0.0001   0.0281   0.0033
24398.8   1.3325   1690.5275   0.0007   0.0000   0.0000   0.0000
24428.1   1.6542   2029.0152   0.0008   0.0000   0.0000   0.0000
24457.4   2.0416   2434.1699   0.0008   0.0000   0.0332   0.0003
```

Now the MCD and the absorption spectra can be plotted with a suitable graphical program, for instance with the Origin program.

Fig. 3.47: Calculated MCD and absorption spectra of $[Fe(CN)_6]^{3-}$ (dash lines) compared to experimental spectra (solid lines).

Addition of Magnetic Fields

The inclusion of the Zeeman contribution into the QDPT procedure allows to obtain the splittings of the magnetic levels in an external magnetic field. The switch for this calculation and the magnetic field strength are defined in the soc subblock of the mrci block. Optionally the wave function decomposition can be printed for `MagneticField_PrintLevel` larger 0. The latter employs the thresh `TPrint` to omit small contributions from the printing:

```
%mrci
  soc
```

(continues on next page)

(continued from previous page)

```
    DoSOC true          #
    DoSSC true          #
    MagneticField true  # default false
    B 1,10,100,1000     # Strengh of the magnetic field in Gauss.
                        # 4000 is the default value

    # Optional printing of the wave function for each
    # magnetic field settings
    MagneticField_PrintLevel 0 # default (disabled)
    TPrint                    1e-3
  end
end
```

Then, the output contains three sets of data of splittings of the magnetic levels with the magnetic field applied parallel to x, y, and z directions:

```
End B (Gauss)  Energy levels (cm-1) and populations for B || x

     1.0   -0.030   0.333   0.012   0.333   0.018   0.333
    10.0   -0.030   0.333   0.012   0.333   0.018   0.333
   100.0   -0.031   0.333   0.012   0.333   0.020   0.333
  1000.0   -0.102   0.333   0.012   0.333   0.091   0.333

B (Gauss)  Energy levels (cm-1) and populations for B || y

     1.0   -0.030   0.333   0.012   0.333   0.018   0.333
    10.0   -0.030   0.333   0.012   0.333   0.018   0.333
   100.0   -0.032   0.333   0.014   0.333   0.018   0.333
  1000.0   -0.105   0.334   0.018   0.333   0.087   0.333

B (Gauss)  Energy levels (cm-1) and populations for B || z

     1.0   -0.030   0.333   0.012   0.333   0.018   0.333
    10.0   -0.030   0.333   0.011   0.333   0.018   0.333
   100.0   -0.030   0.333   0.005   0.333   0.025   0.333
  1000.0   -0.079   0.333  -0.030   0.333   0.108   0.333
```

Here the number in a row represents the strength of the magnetic field (in Gauss), and the following pairs of numbers denote the energy of the magnetic level (in cm^{-1}) with its occupation number. This table can be readily plotted with any suitable graphical program.

Relativistic Picture Change in Douglas-Kroll-Hess SOC and Zeeman Operators

The DKH correction to the SOC operator is implemented in ORCA as a correction to the one-electron part of the SOMF operator. The DKH transformation is performed up to the second order, and the two-electron part in our implementation is left untransformed. However, the electronic density employed for evaluating the SOMF matrix elements is obtained from a scalar relativistic calculation. The inclusion of the DKH correction is controlled by the picturechange key in the rel block:

```
%rel method DKH              # relativistic method
     picturechange 2         # include the DKH correction to SOC
     end
```

The "picturechange" key can be set to 0, 1, and 2 for no picture change, the first order, and the second order DKH transformations of the SOC operator.

With "picturechange" set to 1 or 2 the DKH correction are applied in the first order to the Zeeman operator. This correction has a visible effect on calculated g-tensors for molecules containing third-row and heavier atoms.

X-ray Spectroscopy

Likewise to the CASCI/NEVPT2 computational protocol presented in section *Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS* starting from ORCA 4.2 the MRCI module can be used to compute core excited spectra, namely X-ray absorption (XAS) and resonant inelastic scattering (RIXS) spectra.

As discussed in the case of CASCI/NEVPT2 protocol *Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS* a similar strategy is followed to compute XAS/RIXS spectra within the MRCI module. In principle the XAS/RIXS spectra calculations require two steps:

- In a first step one needs to optimize the valence active space orbitals in the framework of SA-CASSCF calculations, e.g. including valence excited states in the range between 6 to 15 eV.
- In a second step the relevant core orbitals are rotated into the active space and the MRCI problem is solved by saturating the excitation space with singly core-excited electronic configurations using the previously optimized sets of orbitals.
- The core orbitals are also included in the XASMOs definition. The use of this keyword is two fold. At first it effecteively reduces the number of the generated configuration state functions (CSFs) to those that exclusively contain contributions from the defined core orbitals. In the case of RIXS also XES (see below) the specified XASMOs are used to define intermediate or core ionized states.

A representative input for the case of Fe(Cl) $_4$ is provided bellow:

- In the first step one performs a SA-CASSCF calculation for the 5 and 15 quintet and triplet states (FeIICl4.casscf.inp).

```
!cc-PWCVTZ-DK cc-pVTZ/C RIJCOSX SARC/J TightSCF DKH2

%rel
  FiniteNuc true
end

%basis
  newgto Cl "cc-pVTZ-DK" end
  newauxgto Cl "cc-pVTZ/C" end
end

%method FrozenCore FC_NONE
end

%casscf nel   6
  norb   5
  mult   5,3
  nroots 5,15
  switchstep nr
end

* xyz -2 5
Fe  -17.84299991694815     -0.53096694321123      6.09104775508499
Cl  -19.84288422845700      0.31089495619796      7.04101319789001
Cl  -17.84298666758073      0.11868125024595      3.81067954087770
Cl  -17.84301352218429     -2.87052442818457      6.45826391412877
Cl  -15.84311566482982      0.31091516495189      7.04099559201853
*
```

- In a second step the core orbitals are rotated in the active space and the MRCI problem is solved by saturating the excitation space with all the quintet and triplet states that involve single excitations from the core orbitals (FeIICl4-mrci.inp)

```
!MORead CC-PWCVTZ-DK cc-pVTZ/C  RIJCOSX SARC/J TightSCF DKH2

%moinp "FeIICl4-casscf.gbw"
```

(continues on next page)

(continued from previous page)

```
%rel
  FiniteNuc true
end

%method FrozenCore FC_NONE
end

%scf
  rotate { 6,42,90} { 7,43,90} { 8,44,90}  end
end

%basis
  newgto Cl "cc-pVTZ-DK" end
  newauxgto Cl "cc-pVTZ/C" end
end


%casscf
  nel    12
  norb   8
  mult   5,3
  nroots 34,195
  maxiter 1
  switchstep nr
end


%mrci
  CIType MRCI
  intmode fulltrafo
  XASMOs 42, 43, 44
  newblock 5 *
  nroots 34
  excitations cisd
  refs CAS(12,8)
  end
  end
  newblock 3 *
  nroots 195
  excitations cisd
  refs CAS(12,8)
  end
  end
  maxiter 100
  soc
  printlevel 3
  dosoc true
  end
end


* xyz -2 5
Fe  -17.84299991694815     -0.53096694321123      6.09104775508499
Cl  -19.84288422845700      0.31089495619796      7.04101319789001
Cl  -17.84298666758073      0.11868125024595      3.81067954087770
Cl  -17.84301352218429     -2.87052442818457      6.45826391412877
Cl  -15.84311566482982      0.31091516495189      7.04099559201853
*
```

In a similar fashion Multireference Equation of Motion Couple Cluster MR-EOM-CC (see next section) can also be used to compute X-ray spectra. Further information can be found in reference[514]

As it is explicitly described in the respective ROCIS section RIXS spectra can be requested by the following keywords:

```
RIXS    true    # Request RIXS calculation (NoSOC)
RIXSSOC true    # Request RIXS calculation (with SOC)
Elastic true    # Request RIXS calculation (Elastic)
```

Please consult section *Resonant Inelastic Scattering Spectroscopy* for processing and analyzing the generated spectra

Likewise to TDDFT (*Excited States via RPA, CIS, TD-DFT and SF-TDA*) ROCIS (*Excited States with Restricted Open-shell CIS - ROCIS*) and CASSCF (*Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS*) the computed transition densities also in the presence of SOC can be taken beyond the dipole approximation by using the *OPS tool* for details.

1. by performing a multipolar expantion
2. by computing the full semiclassical transition moments

The whole set of spectroscopy tables can be requested with the following commands:

```
%mrci
 DoDipoleLength        true
 DoDipoleVelocity      true
 DecomposeFosc         true
 DoFullSemiclassical   true
end
```

More details can be found in TDDFT (*Excited States via RPA, CIS, TD-DFT and SF-TDA*) ROCIS (*Excited States with Restricted Open-shell CIS - ROCIS*) and CASSCF (*Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS*) sections.

Starting from ORCA 4.2 the previously reported RASCI-XES protocol reference[515], which can compute K_β Mainline XES spectra, can be processed entirely within the ORCA modules. In ORCA 5.0 a similar protocol (CASCI-XES) exist in the CASSCF module (*Core Excited Spectra: CAS-CI/RAS-CI XAS/RIXS*)

- Like above or in the CASCI/NEVPT2 case in a first step one needs to optimize the valence active space orbitals in the framework of SA-CASSCF calculations, e.g. including valence excited states in the range between 6 to 15 eV for the N electron system.
- In a second step the metal 1s and 3p orbitals are rotated in the active space and the 1s MO is defined in the XASMOs list
- Computes the XES spectrum in the RASCI framework for the N-1 electron system in the presence of SOC if the XESSOC keyword for all the states that are dominated by 3p-1s electron decays.

A representative input sequence for the case of Fe(Cl) $_6$ is provided bellow:

As described in reference[515] at first for a CAS(5,5) the excitation space is saturated by the sextet as well as the 24 quartet and the 75 doublet states which are optimized in the SA-CASSCF fashion.

```
X2c x2c-TZVPall x2c/J  def2-TZVP/C


%scf
  MaxDisk 40000
end

%casscf
  nel 5
  norb 5
  mult 6,4,2
  nroots 1,24,75
  shiftup 0.5
  shiftdn 0.5
  trafostep RI
  maxiter 150
end
```

(continues on next page)

(continued from previous page)

```
*xyz -3  6
Fe      0.0000       0.0000     0.000000
Cl      2.478        0.0000     0.000
Cl     -2.478        0.0000     0.000
Cl      0.000005     2.478      0.00000
Cl      0.000005    -2.478     -0.0000
Cl     -0.000       -0.000      2.478
Cl      0.000       -0.0000    -2.478
*
```

In following the 1s and 3p Fe based MOs are rotated in the active space and the XES spectra are computed for the $[Fe(Cl)_6]^+$ system for the 4 septet and 81 quintet states.

```
! X2c x2c-TZVPall x2c/J  def2-TZVP/C MORead AllowRHF
! NormalPrint
! NoLoewdin NoMulliken


%moinp "fecl6_casscf.gbw"


%scf
  MaxDisk 40000
end


%scf
  #Rotate the 1s and 3p orbitals below the SOMOs by using the rotate option
  rotate {0,59,90} {36,60,90} {37,61,90} {38,62,90} end
end

%casscf
  nel 5
  norb 5
  mult 6,4
  nroots 1,24
  shiftup 0.5
  shiftdn 0.5
  trafostep exact
  maxiter 1
end


%mrci citype mrci
 UseIVOs false
 Etol 1e-8
 IntMode fulltrafo
 PrintLevel 3
 newblock 7 *
  excitations none
  nroots 4
  refs ras(12:4 1/5/0 0) end
 end
 newblock 5 *
  excitations none
  nroots 81
  refs ras(12:4 1/5/0 0) end
 end
 XASMOS 59
 soc
  dosoc true
```

(continues on next page)

(continued from previous page)

```
  xessoc true
 end
end


*xyz -2  7
Fe      0.0000       0.0000     0.000000
Cl      2.478        0.0000     0.000
Cl     -2.478        0.0000     0.000
Cl      0.000005     2.478      0.00000
Cl      0.000005    -2.478     -0.0000
Cl     -0.000       -0.000     2.478
Cl      0.000       -0.0000    -2.478
*
```

As a result the X-ray emission spectrum is calculated and the intensities are computed on the basis of the transition electric dipole moments

```
-------------------------------------------------------------------------------
↪---------------------
      SOC CORRECTED   EMISSION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪---------------------
      Transition         Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                          (eV)      (cm-1)    (nm)                 (au**2)    (au)␣
↪     (au)      (au)
-------------------------------------------------------------------------------
↪---------------------
 426-7.0A  -> 420-5.0A   7083.210737 57129953.4     0.2   0.000000203   0.00000  ␣
↪0.00002   0.00002   0.00002
 427-7.0A  -> 420-5.0A   7083.210737 57129953.4     0.2   0.000003409   0.00000  ␣
↪0.00003   0.00014   0.00000
 426-7.0A  -> 419-5.0A   7083.210739 57129953.4     0.2   0.000000626   0.00000  ␣
↪0.00004   0.00004   0.00002
 427-7.0A  -> 419-5.0A   7083.210739 57129953.4     0.2   0.000000169   0.00000  ␣
↪0.00000   0.00001   0.00003
 426-7.0A  -> 418-5.0A   7083.210756 57129953.5     0.2   0.000000867   0.00000  ␣
↪0.00003   0.00006   0.00002
 427-7.0A  -> 418-5.0A   7083.210756 57129953.5     0.2   0.000000228   0.00000  ␣
↪0.00000   0.00000   0.00004
 426-7.0A  -> 417-5.0A   7083.210911 57129954.8     0.2   0.000000824   0.00000  ␣
↪0.00001   0.00003   0.00006
 427-7.0A  -> 417-5.0A   7083.210911 57129954.8     0.2   0.000001296   0.00000  ␣
↪0.00008   0.00004   0.00000
 426-7.0A  -> 416-5.0A   7083.211098 57129956.3     0.2   0.000000159   0.00000  ␣
↪0.00000   0.00002   0.00002
  ...
 428-5.0A  ->  0-5.0A   7159.092436 57741980.6     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
 430-5.0A  ->  0-5.0A   7159.092436 57741980.6     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
 429-5.0A  ->  0-5.0A   7159.092436 57741980.6     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
 431-5.0A  ->  0-5.0A   7159.092436 57741980.6     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
 432-5.0A  ->  0-5.0A   7159.092436 57741980.6     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
-------------------------------------------------------------------------------
↪---------------------
```

The resulted XES spectrum can be visualized by processing the above output file with the `orca_mapspc`

```
orca_mapspc fecl6_xes.out XESSOC -x07000 -x17200 -w4.0 -eV -n10000
```

Note

It is in general not recomended to compute rasci XES spectra on the basis of mrci module.

- In fact K_α: Fe $2p \rightarrow$ Fe $1s$ and K_α: Fe $2p \rightarrow$ Fe $1s$ (Mainline XES) spectra are extreme state demanding and require a large number of releativistic transition densities to be generate which brings a lot of stress to the mrci module.
- A more direct and computationally efficient alternative is provided by the rasci xes protocol via the %casscf module as described in *Core Excited Spectra: CAS-CI/RAS-CI XES*. This in principle provides suitable protocols for the computation of all X-ray emission
- K_α: Fe $2p \rightarrow$ Fe $1s$
- $K_{\beta_{1,3}}$: Fe $3p \rightarrow$ Fe $1s$ (Mainline XES)
- $K_{\beta_{2,5}}$ / $K_{\beta'}$: (Cl $2p$ + Fe $3d$) $\rightarrow$ Fe $1s$ (VtC-XES)

as well the respective resonant XES (RIXS) spectra.

- We also recall that within the limits of their applicability XES and RIXS spectra can also be computed on the basis of Static Ground State DFT (SGS-DFT), see discussion in *Static Ground State DFT (SGS-DFT)* and ROCIS family of methods, see discussion in *Resonant Inelastic Scattering Spectroscopy*.
- A general discussion regarding X-ray spectroscopy experiements and how to approach them on the basis of modern computational spectroscopy methodologies is provided in reference [458]

Hence by processing the above spectra one obtains the exact similar spectrum as presented in Fig. 3.23

3.19.15 Keywords

Caution

The MRCI program is considered legacy.

Simple input keywords for the MRCI module are given in Table 3.52. Note that if these are used, all information about reference spaces, number of roots etc. is taken from the CASSCF module that is assumed to be run in advance. In general, these calculations are of the individually selecting type and are very time consuming. Very many flags can be set and modified for these methods and in general using these methods requires expert users! See the variables `Tsel`, `Tpre` and `Tnat` that define the individual selection process. All of these methods can be used with RI integrals by using `RI-<Method>` etc. However, then the calculations become even more time consuming since integrals are made one- by one on the fly. Non-RI calculations will be pretty much limited to about 200-300 orbitals included in the CI. A reasonably complete list of `%mrci` block keywords and their meaning follows below.

Table 3.52: Simple input keywords for the MRCI module.

Keyword	Description
`MRCI`	Initiates a multireference configuration interaction calculation with single and double excitations
`MRCI+Q`	Same with multireference Davidson correction for unlinked quadruples
`MRACPF`	Average coupled-pair functional
`MRAQCC`	Average quadratic coupled-cluster
`MRDDCI1`	Difference dedicated CI with one degree of freedom
`MRDDCI2`	Same with two degrees of freedom
`MRDDCI3`	Same with three degrees of freedom
`MRDDCI<n>+Q`	MRDDCI with Davidson correction (`n` = 1–3)
`SORCI`	Spectroscopy oriented CI

```
%mrci
       # ------------------------------------------------------------
       # Orbital selection
       # NOTE: The orbitals are used as supplied. Thus, the ORDER of
       # orbitals is critical. Say you have
       #   nact    electrons in the active space
       #   nint    electrons in the internal space
       #   nfrozen electrons
       # * The first nfrozen/2 orbitals will not be included in the CI
       # * The next nint/2 orbitals will be doubly occupied in all
       #   references
       # * the nact electrons are distributed over the,say, mact
       #   orbitals according to the active space definitions.
       # The remaining orbitals are external.
       # IT IS YOUR RESPONSIBILITY THAT THE ORBITAL ORDERING MAKES
       # SENSE!
       # A sensible two-step procedure is:
       #  * generate some orbitals and LOOK AT THEM. Decide which ones
       #    to include in the CI.
       #  * re-read these orbitals with ! MORead NoIter. Perhaps use
       #    the "rotate" feature to reorder the MOs
       #    Then jump right into the CI which is defined in this se-
       #    cond job
       #
       # NOTE: the MRCI module respects the %method FrozenCore settings
       # ------------------------------------------------------------
       Loc  0,0,0
           # Localize orbitals in the internal (first flag), active
           # (second flag) and external space (third flag).
       UseIVOs  false
           # Use improved virtual orbitals in the CI
       # orbital energy window for the frozencore option FC_EWIN
       EWIN -40,1000
       # alternative MO definition
       MORanges  First_internal, Last_Internal, First_active, Last_Active, First-
↪Virtual,Last_virtual
       XASMOs  x1,x2,x3,...   # List of XAS donor MOs (see above)

       # ---------------------------------
       # Method selection
       # ---------------------------------
       CIType   MRCI      # Multireference CI (default)
                MRDDCI1   # Difference dedicated CI 1-degree of freedom
                MRDDCI2   # Difference dedicated CI 2-degrees of freedom
                MRDDCI3   # Difference dedicated CI 3-degrees of freedom
                MRACPF    # Average coupled-pair functional
                MRACPF2   # Modified version of ACPF
                MRACPF2a  # A slightly modified version of ACPF-2a
                MRAQCC    # Average quadratic coupled-cluster
                MRCEPA_R  # Multireference CEPA due to Ruttink
                MRCEPA_0  # CEPA-0 approximation
                SORCI     # Spectroscopy oriented CI
                SORCP     # Spectroscopy oriented couplet pair approx.
                MRMP2     # Multireference Moeller-Plesset at second order
                MRMP3     # Multireference Moeller-Plesset at third order
                MRMP4     # Multireference Moeller-Plesset at fourth order
                          # but keeping only singles and doubles relative to
                          # the reference configurations.

       # ---------------------------------
       # Selection thresholds
       # ---------------------------------
```

(continues on next page)

(continued from previous page)

```
Tsel    1e-6   # Selection threshold for inclusion in the CI based
               # 2nd order MP perturbation theory <0|H|I>/DE(MP)
Tpre    1e-4   # Selection of configurations in the reference space
               # after the initial diagonalization of the reference
               # space only configurations with a weight large>Tpre
               # to any root are included
AllSingles   false
               # include ALL SINGLES in the CI. Default is now TRUE!!!

# perturbative estimate of the effect of the rejected configurations
EunselOpt  0   # no correction
           1   # based on the overlap with the 0th order wavefunction
           2   # calculation with the relaxed reference space
               # coefficients. This is the most accurate and only
               # slightly more expensive

# For CIType=MRCI,MRDDCI and SORCI the approximate correction for
# higher excitations
DavidsonOpt  Davidson1  # default
             Davidson2  # modified version
             Siegbahn   # Siegbahn's approximation
             Pople      # Pople's approximation

# For MRACPF,MRACPF2,MRAQCC and SORCP
NelCorr      0
   # Number of electrons used for computing the average coupled-
   # pair correction.
   # =0 : set equal to ALL electrons in the CI
   # =-1: set equal to all ACTIVE SPACE electrons
   # =-2: set equal to ACTIVE SPACE electrons IF inactive doubles
   #      are excluded (as in MRDDCI)
   # >0 : set equal to user defined input value
LinearResponse false
   # Use ground state correlation energy to compute the shift for
   # higher roots (not recommended)

# ---------------------------------
# Natural Orbital Iterations
# ---------------------------------
NatOrbIters    0 # default
   # number of average natural orbital iterations
Tnat        1e-4
   # cutoff of natural orbitals. NOs with an occupation number less
   # then Tnat will not be included in the next iteration
   # Also, orbitals with occupation number closer than Tnat to 2.0
   # will be frozen in the next iteration
Tnat2       -1
   # if chosen >0 then Tnat2 is the threshold for freezing the
   # almost doubly occupied orbitals. Otherwise it is set equal
   # to Tnat

# ----------------------------------
# Additional flags and algorithmic
# details
# ----------------------------------
PrintLevel   2  # default. Values between 1 and 4 are possible

DoDDCIMP2  false
   # for DDCI calculations: if set to true the program computes
   # a MP2 like correction for the effect of inactive double
   # excitations which are not explicitly included in the CI. This
```

(continues on next page)

(continued from previous page)

```
          # is necessary if you compare molecules at different geometries
          # or compute potential energy surfaces.
       # ----------------------------------
       # The SORCP model
       # ----------------------------------
       CIType_in  # First step CIType
       CIType_fi  # Second step CIType
       Exc_in     # First step excitation scheme
       Exc_fi     # Second step excitation scheme
       Tsel_in    # First step Tsel
       Tsel_fi    # Second step Tsel
       Tpre_in    # First step Tsel
       Tpre_fi    # Second step Tpre

         # Thus, the SORCI model corresponds to CIType=SORCP with
         # CIType_in MRCI  CIType_fi MRCI
         # Exc_in    DDCI2 Cexc_fi   DDCI3
         # Tsel_in   1e-5  Tsel_fi   1e-5
         # Tpre_in   1e-2  Tpre_fi   1e-2

       # ----------------------------------
       # Multirerence perturbation theory
       # ----------------------------------
       MRPT_b  0.02   # Intruder state avoidance PT after Hirao (default 0.0)
                      # with this flag individual intruders are shifted away to
                      # to some extent from the reference space
       MRPT_shift 0.3 # Level shift introduced by Roos which shifts the entire
                      # excited manifold away in order to avoid intruder states.
                      # A correction is applied afterwards but results do depend
                      # on this (arbitrary) value to some extent.
       H0Opt  projected  # use an off-diagonal definition of H0
              Diagonal   # use a diagonal definition of H0 (much faster but maybe
                         # a little less reliable
       Partitioning MP   # Moeller plesset partitioning
                    EN   # Epstein-Nesbet partitioning (not recommended)
                    RE   # Fink's partitioning
       Fopt  Standard    # Standard definition of MR Fock operators
             G3          # uses Anderson's g3 correction also used in CASPT2
       UsePartialTrafo   true/false    # speedups MRMP2

       #---------------------------------------
       # restrict reference configurations
       #---------------------------------------
       RejectInvalidRefs true # by default reference CSFs are restricted
                              # to target spin and spatial symmetry

       # ======================================
       # Definitions of blocks of the CI Matrix
       # ======================================
       NewBlock 2 *    # generate a Block with doublet(=2) multiplicity
         Nroots  1     # number of roots to be generated
         Excitations cis   # CI with single excitations
                     cid   # CI with double excitations
                     cisd  # CI with single and double excitations
                     ddci1 # DDCI list with one degree of freedom
                     ddci2 # DDCI list with two degrees of freedom
                     ddci3 # DDCI list with three degrees of freedom
         Flags[_class_] 0 or 1
                  # Turn excitation classes on or off individually
                  # ``s'' stands for any SOMO, ``i'',``j'' for internal orbitals␣
↪and
```

(continues on next page)

(continued from previous page)

```
                  # ``a'',``b'' for external orbitals
                  #  Singles _class_ = ss, sa, is, ia
                  #  Doubles _class_ = ijss, ijsa, ijab,
                  #                    isss, issa, isab,
                  #                    ssss, sssa, ssab
                  # ``Flags'' takes priority over ``Excitations''. In fact␣
↪``Excitations''
                  # does nothing but to set ``Flags''. So, you can use␣
↪``Excitations''
                  # to provide initial values for ``Flags'' and then modify them
                  # with subsequent ``Flags'' assignments
        refs
              #
              # First choice - complete active space
              #
              CAS(nel,norb)  # CAS-CI reference with nel electrons in
                             # Norb orbitals
              #
              # Second choice - restricted active space
              #
              RAS(nel: m1 h/ m2 / m3 p)
                             # RAS-reference with nel electrons
                    # m1= number orbitals in RAS-1
                    # h = max. number of holes in RAS-1
                    # m2= number of orbitals in RAS-2 (any number of
                    #     electrons or holes)
                    # m3= number of orbitals in RAS-3
                    # p = max. number of particles in RAS-3
              #
              # Third choice - individually defined configurations
              #
               \{ 2 0 1 0\}
               \{ 1 1 1 0\}
                 etc.
                 # define as many configurations as you want. Doubly occupied MOs
                 # singly occupied MOs and empty MOs. Important notes:
                 #  a) the number of electrons must be the same in all references
                 #  b) the number of orbitals is determined from the number of
                 #     definitions. Thus, in the example above we have three␣
↪active
                 #     electrons and four active orbitals despite the fact that␣
↪the
                 #     highest orbital is not occupied in any reference.
                 #  The program determines the internal, active and external␣
↪spaces
                 #  automatically from the number of active electrons and orbitals
              end
        end
        # there can be as many blocks as you want!!!

      # ----------------------------------
      # Density matrix generation flags
      #  First Key= State densities <I|D|I>
      #     =0: none
      #     =1: Ground state only (lowest root of all blocks; Electron only)
      #     =2: Ground state only (Electron and spin density)
      #     =3: Lowest root from each block (Electron density)
      #     =4: Lowest root from each block (Electron and spin density)
      #     =5: All states (Electron density)
      #     =6: All states (Electron and spin density)
      #  Second Key= Transition densities <I|D|J>
```

(continues on next page)

(continued from previous page)

```
      #     needed for all transition intensities, g-tensor etc
      #     =0: none
      #     =1: from the ground state into all excited states (el)
      #     =2: from the ground state into all excited states (el+spin)
      #     =3: from all lowest states into all excited states (el)
      #     =4: from all lowest states into all excited states (el+spin)
      #     =5: all state pairs (el)
      #     =6: all state pairs (el+spin)
      # Note that for perturbation theory the density is computed as
      # an expectation value over the first (second) order wavefunction.
      # which is renormalized for this purpose
      # ----------------------------------
      Densities 1,1

      # ----------------------------------
      # Complete printing of the wavefunction
      # ----------------------------------
      PrintWF    1      # CFG based printing (default)
                 det    # Determinant based wavefunction printing
      TPrintWF   3e-3   # Threshold for the printing of the CFGs/Dets

      # ----------------------------------
      # Algorithm for the solver
      # ----------------------------------
      Solver   Diag   # Davidson like solver
               DIIS   # DIIS like solver
        # both solvers have their pros and cons. The DIIS may converge
        # better or use less time since it only recomputes the vectors that
        # have not yet converged; The DIIS may be less sensitive to root flipping
        # effects but occasionally it converges poorly and states of the same
        # symmetry are occasionally a little problematic
        # For perturbation theory DIIS is always used.
      # For both solvers
      MaxIter   100  # the maximum number of iterations
      Etol      1e-6 # convergence tolerance for energies in Eh
      Rtol      1e-6 # convergence tolerance for residual

      # For Solver=Diag (Davidson solver)
      Otol      1e-16 # Orthogonality threshold for Schmidt process
      NGuessMat 512   # Dimension of the guess matrix 512x512
                      # be used to compute the initial guess of the actual MRCI␣
↪calculation
      NGuessMatRefCI 512 # Dimension of the guess matrix
                         # for the reference CI

      MaxDim       3  # Davidson expansion space = MaxDim * NRoots
      # For the Solver=DIIS. Particularly recommended for anything else but
      # straightforward CI and also for calculations in direct2 mode!
      MaxDIIS       5 # Maximum number of old vectors to be used in DIIS
      RelaxRefs  true # Relax reference space coefficients in the CI or
                      # freeze them to their zeroth order values
      LevelShift  0.4 # Level Shift for stabilizing the DIIS procedure

      # ----------------------------------
      # RI Approximation
      # ----------------------------------
      IntMode RITrafo #Use RI integrals
              FullTrafo #No RI (default)

      # ----------------------------------
      # Integral storage, memory and files
```

(continues on next page)

(continued from previous page)

```
      # ----------------------------------
      FourIndexInts false (default)
                    True
         # Store ALL four index integrals over Mos in main memory
         # only possible for relatively small systems, perhaps up
         # to 150-200 MOs included in the CI
      MaxMemInt     256
         # Maximum amount of core memory devoted to the storage of
         # integrals. If NOT all three index integrals fit into main
         # memory the program fails
      MaxMemVec      16
         # Maximum amount of memory used for section of the trial and
         # sigma vectors. This is not a particularly critical variable
      KeepFiles    false
         # Keep integrals and CI program input file (.mrciinp). Then
         # you can manually edit the .mrciinp file which is a standard
         # ASCII file and run the MRCI program directly. The only thing
         # you cannot change is the orbital window.
      end
```

3.20 Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC)

The Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC) methodology [516, 517, 518, 519, 520, 521] has been implemented in ORCA. The strength of the MR-EOM-CC methodology lies in its ability to calculate many excited states from a single state-averaged CASSCF solution, for which only a single set of amplitudes needs to be solved and the final transformed Hamiltonian is diagonalized over a small manifold of excited states only through an uncontracted MRCI problem. Hence, a given MR-EOM-CC calculation involves three steps, performed by three separate modules in ORCA :

1. a state-averaged CASSCF calculation (CASSCF module),
2. the solution of amplitude equations and the calculation of the elements of the similarity transformed Hamiltonians (MDCI module),
3. and the uncontracted MRCI diagonalization of the final similarity transformed Hamiltonian (MRCI module).

The current implementation allows for MR-EOM-T|T†-h-v, MR-EOM-T|T†|SXD-h-v and MR-EOM-T|T†|SXD|U-h-v calculations. A more detailed description of these methods and the available input parameters will be given in Sec. Section 3.20.4. We also note that the theoretical details underlying these methods can be found in Ref. [521]. In Sec. Section 3.20.4, we will discuss a strategy for the selection of the state-averaged CAS and other steps for setting up an MR-EOM-CC calculation in more detail. Furthermore, we will discuss how spin-orbit coupling effects can be included in MR-EOM-CC calculations, a projection scheme to aid with convergence difficulties in the iteration of the T amplitude equations, an orbital selection scheme to reduce the size of the inactive core and virtual subspaces in the calculation of excitation energies and a strategy for obtaining nearly size-consistent results in MR-EOM-CC The purpose of this section is simply to provide a simple example which illustrates the most basic usage of the MR-EOM-CC implementation in ORCA.

3.20.1 A Simple MR-EOM-CC Calculation

Let us consider an MR-EOM-T|T†|SXD|U-h-v calculation on formaldehyde. An MR-EOM-T|T†|SXD|U-h-v calculation is specified via the `MR-EOM` keyword along with the specification of a state-averaged CASSCF calculation (i.e. CASSCF(nel, norb) calculation with the number of roots of each multiplicity to be included in the state-averaging for the reference state) and the number of desired roots in each multiplicity block for the final MRCI diagonalization. We note that the CASSCF module is described in section Section 3.13 and that a description of the MRCI module is given in section Section 3.19. Here, we have a state-averaged CAS(6,4) calculation, comprised of 3 singlets and 3 triplets and we request 6 singlet roots and 6 triplet roots in our final MRCI diagonalization (i.e. the roots to be computed in the MR-EOM-T|T†|SXD|U-h-v calculation):

```
!MR-EOM def2-TZVP VeryTightSCF

%casscf  # reference state
 nel 6
 norb 4
 mult 1,3
 nroots 3,3
end

%mdci
  STol 1e-7
end

%mrci  # final roots
  newblock 1 *
    nroots 6
    refs cas(6,4) end
  end
  newblock 3 *
    nroots 6
    refs cas(6,4) end
  end
end


* xyz 0 1
 H        0.000000        0.934473       -0.588078
 H        0.000000       -0.934473       -0.588078
 C        0.000000        0.000000        0.000000
 O        0.000000        0.000000        1.221104
*
```

One can alternatively perform an MR-EOM-T|T†-h-v or MR-EOM-T|T†|SXD-h-v calculation by replacing the `MR-EOM` keyword, in the first line of the input above, by `MR-EOM-T|Td` or `MR-EOM-T|Td|SXD`, respectively. Namely, replacing the first line of the input above with

```
!MR-EOM-T|Td def2-TZVP VeryTightSCF
```

runs the MR-EOM-T|T†-h-v calculation, while

```
!MR-EOM-T|Td|SXD def2-TZVP VeryTightSCF
```

runs the MR-EOM-T|T†|SXD-h-v calculation.

The final MRCI diagonalization manifold includes 2h1p, 1h1p, 2h, 1h and 1p excitations in MR-EOM-T|T†-h-v calculations, 2h, 1p and 1h excitations in MR-EOM-T|T†|SXD-h-v calculations and 1h and 1p excitations in MR-EOM-T|T†|SXD|U-h-v calculations. Note that in the `%mdci` block, we have set the convergence tolerance (`STol`) for the residual equations for the amplitudes to 10^{-7}, as this default value is overwritten with the usage of the `TightSCF`, `VeryTightSCF`, etc. keywords. It is always important to inspect the values of the largest T, S (here, we use S to denote the entire set of S, X and D amplitudes) and U amplitudes. If there are amplitudes that are large (absolute values > 0.15), the calculated results should be regarded with suspicion. For the above calculation, we obtain:

```
--------------------
LARGEST T AMPLITUDES
--------------------
   8-> 13   8-> 13       0.060331
   4-> 17   4-> 17       0.029905
   8->  9   8->  9       0.028160
   8-> 16   8-> 16       0.027266
   6-> 20   6-> 20       0.025885
   8-> 21   8-> 21       0.025308
   4-> 16   4-> 16       0.024803
   8-> 12   8-> 12       0.023915
   5-> 18   5-> 18       0.023553
   8-> 23   8-> 23       0.023384
   3-> 16   3-> 16       0.023182
   7-> 19   7-> 19       0.023043
   8-> 13   4-> 11       0.022010
   3-> 19   3-> 19       0.021987
   8-> 16   8->  9       0.021230
   8->  9   8-> 16       0.021230
```

for the T amplitudes,

```
--------------------
LARGEST S AMPLITUDES
--------------------
   4->  8   8-> 11       0.074048
   3->  8   8->  9       0.064886
   4->  5   5-> 11       0.045479
   3->  8   8-> 16       0.042657
   4->  7   7-> 11       0.042598
   4->  5   5-> 17       0.042076
   4->  5   8-> 11       0.039958
   4->  8   8-> 17       0.037532
   3->  5   8->  9       0.035907
   4->  7   7-> 17       0.035767
   2->  6   6-> 19       0.034148
   3->  5   5-> 10       0.033339
   2->  6   6-> 10       0.032691
   4->  6   6-> 11       0.032181
   8->  8   3-> 16       0.031775
   2->  7   7-> 22       0.031238
```

for the S amplitudes, and

```
--------------------
LARGEST U AMPLITUDES
--------------------
   3->  8   3->  8       0.026128
   3->  8   3->  5       0.007683
   2->  8   2->  8       0.006182
   3->  8   2->  5       0.006154
   2->  8   3->  5       0.004954
   3->  5   3->  5       0.004677
   4->  8   4->  8       0.003989
   3->  8   2->  8       0.002040
   2->  8   3->  8       0.002040
   2->  8   2->  5       0.001818
   4->  8   4->  5       0.001173
   2->  5   2->  5       0.001107
   4->  5   4->  5       0.000714
   3->  7   3->  7       0.000607
   3->  6   3->  6       0.000521
```

(continues on next page)

(continued from previous page)

```
   2->  5   3->  5       0.000365
```

for the U amplitudes. Hence, one can see that there are no unusually large amplitudes for this calculation. **We note that there can be convergence issues with the T amplitude iterations and that in such cases, the flag:**

```
DoSingularPT true
```

should be added to the `%mdci` block. The convergence issues are caused by the presence of nearly singular T_2 amplitudes and setting the `DoSingularPT` flag to `true` activates a procedure which projects out the offending amplitudes (in each iteration) and replaces them by suitable perturbative amplitudes. For more information, see the examples in section Section 3.20.4.

After the computation of the amplitudes and the elements of the similarity transformed Hamiltonians, within the MDCI module, the calculation enters the MRCI module. For a complete, step by step description of the output of an MRCI calculation, we refer the reader to the example described in section Section 3.19.4. Let us first focus on the results for the singlet states (`CI-BLOCK 1`). Following the convergence of the Davidson diagonalization (default) or DIIS procedure, the following results of the MRCI calculation for the singlet states are printed:

```
----------
CI-RESULTS
----------

The threshold for printing is  0.30 percent
The weights of configurations will be printed. The weights are summed over
all CSFs that belong to a given configuration before printing

STATE   0:  Energy=   -114.321368425 Eh RefWeight=  0.9781  0.00 eV      0.0 cm**-1
      0.0137  : h---h---[0222]
      0.0756  : h---h---[1221]
      0.8879  : h---h---[2220]
STATE   1:  Energy=   -114.176866027 Eh RefWeight=  0.9765  3.93 eV  31714.6 cm**-1
      0.0039  : h---h---[1122]
      0.9726  : h---h---[2121]
      0.0071  : h---h  4[1222]
      0.0085  : h---h  4[2221]
STATE   2:  Energy=   -113.988050555 Eh RefWeight=  0.9774  9.07 eV  73154.8 cm**-1
      0.0044  : h---h---[1212]
      0.9730  : h---h---[2211]
      0.0063  : h---h  3[1222]
      0.0041  : h---h  3[2221]
STATE   3:  Energy=   -113.963862283 Eh RefWeight=  0.8810  9.73 eV  78463.5 cm**-1
      0.7459  : h---h---[1221]
      0.0807  : h---h---[2022]
      0.0533  : h---h---[2220]
      0.0228  : h---h  4[2122]
      0.0034  : h---h---[1220]p13
      0.0072  : h---h---[1220]p18
      0.0236  : h---h---[2120]p11
      0.0148  : h---h---[2120]p14
      0.0069  : h---h---[2120]p17
      0.0056  : h---h---[2120]p20
      0.0098  : h---h---[2210]p19
STATE   4:  Energy=   -113.931144468 Eh RefWeight=  0.0003 10.62 eV  85644.3 cm**-1
      0.0045  : h---h---[0122]p9
      0.0089  : h---h---[1121]p9
      0.9333  : h---h---[2120]p9
      0.0243  : h---h---[2120]p10
      0.0080  : h---h---[2120]p12
      0.0113  : h---h---[2120]p16
STATE   5:  Energy=   -113.929056780 Eh RefWeight=  0.6857 10.68 eV  86102.5 cm**-1
      0.0061  : h---h---[0222]
```

(continues on next page)

(continued from previous page)

```
      0.0918  : h---h---[1221]
      0.5784  : h---h---[2022]
      0.0048  : h---h---[2202]
      0.0047  : h---h---[2220]
      0.2905  : h---h  4[2122]
      0.0045  : h---h---[2021]p13
```

For each state, the total energy is given in E_h; the weight of the reference configurations (`RefWeight`) in the given state is provided, and the energy differences from the lowest lying state are given in eV and cm^{-1}. Also, in each case, the weights and a description of the configurations which contribute most strongly to the given state are also provided. See section Section 3.19.4 for a discussion of the notation that is used for the description of the various configurations. To avoid confusion, we note that in the literature concerning the MR-EOM methodology [517, 518, 519, 520, 521, 522, 523], the term "%active" is used to denote the reference weight multiplied by 100%. In general, `RefWeight` should be > 0.9, such that the states are dominated by reference space configurations. This criterion is satisfied for the first three states and the reference weight of the fourth state is sufficiently close to 0.9. **However, the reference weights of the two higher lying states (especially state 4) are too small and these states should be discarded as the resulting energies will be inaccurate (i.e. states with significant contributions from configurations outside the reference space cannot be treated accurately)** .

In the case of the triplet states (`CI-BLOCK 2`), we obtain the following results:

```
----------
CI-RESULTS
----------

The threshold for printing is  0.30 percent
The weights of configurations will be printed. The weights are summed over
all CSFs that belong to a given configuration before printing

STATE   0:  Energy=   -114.190840989 Eh RefWeight=  0.9693  0.00 eV      0.0 cm**-1
      0.9691  : h---h---[2121]
      0.0079  : h---h  4[1222]
      0.0115  : h---h  4[2221]
STATE   1:  Energy=   -114.106733017 Eh RefWeight=  0.9941  2.29 eV  18459.6 cm**-1
      0.9941  : h---h---[1221]
STATE   2:  Energy=   -114.015150051 Eh RefWeight=  0.9787  4.78 eV  38559.7 cm**-1
      0.9786  : h---h---[2211]
      0.0050  : h---h  3[1222]
STATE   3:  Energy=   -113.939299674 Eh RefWeight=  0.0006  6.84 eV  55206.9 cm**-1
      0.0044  : h---h---[0122]p9
      0.0084  : h---h---[1121]p9
      0.9419  : h---h---[2120]p9
      0.0131  : h---h---[2120]p10
      0.0043  : h---h---[2120]p12
      0.0173  : h---h---[2120]p16
STATE   4:  Energy=   -113.925571130 Eh RefWeight=  0.4017  7.22 eV  58220.0 cm**-1
      0.3863  : h---h---[1122]
      0.0154  : h---h---[2121]
      0.1722  : h---h  4[1222]
      0.4098  : h---h  4[2221]
      0.0045  : h---h---[2120]p13
STATE   5:  Energy=   -113.910479339 Eh RefWeight=  0.0009  7.63 eV  61532.3 cm**-1
      0.0088  : h---h---[0122]p10
      0.0030  : h---h---[1121]p10
      0.0120  : h---h---[2120]p9
      0.9408  : h---h---[2120]p10
      0.0106  : h---h---[2120]p16
      0.0112  : h---h---[2120]p19
```

Here, we see that the first three states have reference weights which are > 0.9, while the reference weights of the final three states are well below that threshold. Hence, the latter three states should be discarded from any meaningful

analysis.

Following the printing of the CI results for the final CI block, the states are ordered according to increasing energy and the vertical transition energies are printed:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is   -114.321368425 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    -1    0    0      0.000      0.000       0.0
  1    3    -1    0    1    130.527      3.552   28647.5
  2    1    -1    1    0    144.502      3.932   31714.6
  3    3    -1    1    1    214.635      5.841   47107.0
  4    3    -1    2    1    306.218      8.333   67207.2
  5    1    -1    2    0    333.318      9.070   73154.8
  6    1    -1    3    0    357.506      9.728   78463.5
  7    3    -1    3    1    382.069     10.397   83854.4
  8    1    -1    4    0    390.224     10.619   85644.3
  9    1    -1    5    0    392.312     10.675   86102.5
 10    3    -1    4    1    395.797     10.770   86867.5
 11    3    -1    5    1    410.889     11.181   90179.7
```

Furthermore, following the generation of the (approximate) densities, the absorption and CD spectra are printed:

```
=========================================
MR-EOM-CC  Non Relativistic Properties
=========================================

-------------------------------------------------------------------------------
-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX
  DY        DZ
                      (eV)      (cm-1)     (nm)                 (au**2)    (au)
 (au)       (au)
-------------------------------------------------------------------------------
-----------------
  0-1A  ->  1-1A    3.932110   31714.6   315.3   0.000000000   0.00000  -0.00000  -
0.00000   0.00000
  0-1A  ->  2-1A    9.070040   73154.8   136.7   0.002137450   0.00962   0.09808  -
0.00000  -0.00000
  0-1A  ->  3-1A    9.728237   78463.5   127.4   0.157495738   0.66081  -0.00000
0.00000  -0.81290
  0-1A  ->  4-1A   10.618534   85644.3   116.8   0.025353906   0.09746  -0.00000  -
0.31218  -0.00000
  0-1A  ->  5-1A   10.675343   86102.5   116.1   0.024673667   0.09434  -0.00000  -
0.00000   0.30715

-------------------------------------------------------------------------------
-------
                     CD SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
-------
     Transition      Energy     Energy  Wavelength    R        MX        MY
MZ
                      (eV)      (cm-1)     (nm)    (1e40*cgs)   (au)      (au)
(au)
-------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
↪-------
  0-1A  ->  1-1A    3.932110   31714.6   315.3   -0.00000   0.00000   0.00000   0.
↪56273
  0-1A  ->  2-1A    9.070040   73154.8   136.7   -0.00000   0.00000  -0.74486   0.
↪00000
  0-1A  ->  3-1A    9.728237   78463.5   127.4   -0.00000  -0.00000  -0.00000  -0.
↪00000
  0-1A  ->  4-1A   10.618534   85644.3   116.8    0.00000   0.35898   0.00000  -0.
↪00000
  0-1A  ->  5-1A   10.675343   86102.5   116.1    0.00000   0.00000  -0.00000  -0.
↪00000
```

Warning

- It is important to note that the transition moments and oscillator strengths (and state dipole moments) have been blindly computed by the MRCI module and currently, no effort has been made to include the effects of the various similarity transformations in the evaluation of these quantities. Hence these quantities are only approximate and should only be used as a qualitative aid to determine which states are dipole allowed or forbidden. Furthermore, since the calculated densities are approximate, so are the results of the population analysis that are printed before the absorption and CD spectra.
- While both the CASSCF and MRCI modules can make use of spatial point-group symmetry to some extent, the MR-EOM-CC implementation is currently limited to calculations in C_1 symmetry.

3.20.2 Capabilities

The MR-EOM-CC methodology can be used to calculate a desired number of states for both closed- and open-shell systems from a single state-averaged CASSCF solution. Currently, the approach is limited to serial calculations and to smaller systems in smaller active spaces. One should be aware that in the most cost-effective MR-EOM-T|T†|SXD|U-h-v approach (i.e. the smallest diagonalization manifold), an MRCI diagonalization is performed over all 1h and 1p excited configurations out of the CAS, which will inevitably limit the size of the initial CAS which can be used. We have also implemented an orbital selection scheme which can be used to reduce the size of the inactive core and virtual subspaces in the calculation of excitation energies, and this can be employed to extend the applicability of the approach to larger systems. The current implementation can also be used in conjunction with the spin-orbit coupling submodule (*Properties Calculation Using the SOC Submodule*) of the MRCI module to calculate spin-orbit coupling effects in MR-EOM-CC calculations to first order. These and other features of the current implementation will be discussed in *Details on the Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC) Theory*.

3.20.3 Perturbative MR-EOM-CCPT

The MR-EOM-CC family of methods now also features an almost fully perturbative approach called MR-EOM-T [524]. This method shares the features of the MR-EOM-CC parent method while using non-iterative perturbative estimates for the $\hat{T}$ and $\hat{S}, \hat{X}, \hat{D}$ amplitudes. This slightly reduces the accuracy compared to iterative MR-EOM-CC while reducing runtime. Furthermore, convergence issues due to nearly singular $\hat{T}$ and $\hat{S}, \hat{X}, \hat{D}$ amplitudes cannot occur anymore.

This method can be invoked by adding the keyword `DoMREOM_MRPT True` to the `%mdci` block.

3.20.4 Details on the Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC) Theory

In analogy with single reference EOM-CC and STEOM-CC (see sections Section 5.9 and Section 5.10), Multireference Equation of Motion Coupled-Cluster (MR-EOM-CC) theory [516, 517, 518, 519, 520, 521] can be viewed a transform and diagonalize approach to molecular electronic structure theory. An MR-EOM-CC calculation involves a single state-averaged CASSCF calculation, incorporating a few low-lying states and the solution of a single set of cluster amplitudes, which define a sequence of similarity-transformed Hamiltonians. The ultimate goal of these many-body transformations is to effectively decouple the CAS configurations from important excited configurations (e.g., 2p2h, 2p1h, 1p1h, etc.) which comprise the first-order interacting space. Through the definition of suitable cluster operators, in each of the transformations, most of these excitations can be included in an internally contracted fashion. Hence, the resulting final transformed Hamiltonian can be diagonalized over a small subspace of the original first-order interacting space to gain access to many electronic states. As discussed in section Section 3.20, the MR-EOM-CC implementation in ORCA therefore makes use of the CASSCF module (to obtain the state-averaged CASSCF reference), the MDCI module for the solution of the amplitude equations and the calculation of the elements of the various similarity transformed Hamiltonians and the MRCI module for the diagonalization of the final transformed Hamiltonian. Some desirable features of this methodology are:

- Many states can be obtained through the diagonalization of a similarity transformed Hamiltonian over a compact diagonalization manifold (e.g. the final diagonalization space in MR-EOM-T|T†|SXD|U only includes the CAS configurations and 1h and 1p configurations).
- Only a single state-averaged CASSCF calculation and the solution of a single set of amplitudes is required to define the final similarity transformed Hamiltonian and the results are typically quite insensitive to the precise definition of the CAS (only a few low-lying multiplets need to be included in the state-averaging)
- The MR-EOM-CC approach is rigorously invariant to rotations of the orbitals in the inactive, active and virtual subspaces, and it preserves both spin and spatial symmetry.

Table 3.53: The details of the various MR-EOM-CC transformations that are considered in the ORCA implementation of MR-EOM-CC. The equations for the operator components and the residual equations which determine the corresponding amplitudes also appear in the Table. Note that we use the usual (Einstein) convention that repeated indices are summed over.

Name	Transformation	Operators	Operator Components	Residual Equation
T	$\widehat{\overline{H}} = e^{-\widehat{T}}\widehat{H}e^{\widehat{T}}$	$\widehat{T} = \widehat{T}_1 + \widehat{T}_2$	$\widehat{T}_1 = t_i^a \widehat{E}_i^a$	$R_i^a = \sum_{\mathfrak{m}} w_{\mathfrak{m}} \langle \Phi_{\mathfrak{m}} \vert \widehat{E}_a^i \widehat{\overline{H}} \vert \Phi_{\mathfrak{m}} \rangle$
	$= \overline{h}_0 + \overline{h}_q^p \{\widehat{E}_q^p\} + \overline{h}_{rs}^{pq} \{\widehat{E}_{rs}^{pq}\} + \ldots$		$\widehat{T}_2 = \frac{1}{2} t_{ij}^{ab} \widehat{E}_{ij}^{ab}$	$R_{ij}^{ab} = \overline{h}_{ij}^{ab}$
T†	$\widehat{\overline{H}}_2 e^{-\widehat{T}^\dagger}$	$\widehat{T}^\dagger = \widehat{T}_1^\dagger + \widehat{T}_2^\dagger$	$\widehat{T}_1^\dagger = t_a^i \widehat{E}_a^i$	None (*i.e.* set $t_a^i \approx t_i^a$)
	$= \widetilde{h}_0 + \widetilde{h}_q^p \{\widehat{E}_q^p\} + \widetilde{h}_{rs}^{pq} \{\widehat{E}_{rs}^{pq}\} + \ldots$		$\widehat{T}_2^\dagger = \frac{1}{2} t_{ab}^{ij} \widehat{E}_{ab}^{ij}$	None (*i.e.* set $t_{ab}^{ij} \approx t_{ij}^{ab}$)
SXD	$\widehat{\overline{F}} = \{e^{\widehat{S}+\widehat{X}+\widehat{D}}\}^{-1} \widehat{\mathcal{H}}_2 \{e^{\widehat{S}+\widehat{X}+\widehat{D}}\}$	$\widehat{S} = \widehat{S}_2$	$\widehat{S}_2 = s_{i'j'}^{aw} \widehat{E}_{i'j'}^{aw}$	$R_{i'j'}^{aw} = \overline{f}_{i'j'}^{aw}$
	$= \overline{f}_0 + \overline{f}_q^p \{\widehat{E}_q^p\} + \overline{f}_{rs}^{pq} \{\widehat{E}_{rs}^{pq}\} + \ldots$	$\widehat{X} = \widehat{X}_2$	$\widehat{X}_2 = x_{i'x}^{aw} \widehat{E}_{i'x}^{aw}$	$R_{i'x}^{aw} = \overline{f}_{,i'x}^{aw}$
		$\widehat{D} = \widehat{D}_2$	$\widehat{D}_2 = d_{xi'}^{aw} \widehat{E}_{xi'}^{aw}$	$R_{xi'}^{aw} = \overline{f}_{xi'}^{aw}$
U	$\widehat{G} = e^{-\widehat{U}} \widehat{\overline{F}}_2 e^{\widehat{U}}$	$\widehat{U} = \widehat{U}_2$	$\widehat{U}_2 = u_{i'j'}^{wx} \widehat{E}_{i'j'}^{wx}$	$R_{i'j'}^{wx} = g_{i'j'}^{wx}$
	$= g_0 + g_q^p \{\widehat{E}_q^p\} + g_{rs}^{pq} \{\widehat{E}_{rs}^{pq}\} + \ldots$			

As the details concerning the MR-EOM-CC methodology are rather involved, we refer the interested reader to Refs. [516, 517, 518, 519, 520, 521] for a more detailed discussion. Note that the details concerning the implementation of MR-EOM-CC in ORCA can be found in Refs. [520] and [521]. In the following discussion, we note that general spatial orbitals p, q, r, s, which comprise the molecular orbital basis, are partitioned into (doubly occu-

pied) inactive core orbitals i', j', k', l', occupied orbitals i, j, k, l (i.e. the union of the inactive core and active orbital subspaces), active orbitals w, x, y, z and virtual orbitals a, b, c, d. In general, the many-body similarity transformations assume the general form

$$\widehat{G} = \{e^{\widehat{Y}}\}^{-1}\widehat{\overline{H}}_2\{e^{\widehat{Y}}\} = g_0 + \sum_{p,q} g_q^p\{\widehat{E}_q^p\} + \sum_{p,q,r,s} g_{rs}^{pq}\{\widehat{E}_{rs}^{pq}\} + \ldots,$$

in which $\widehat{Y}$ is a cluster operator and $\widehat{\overline{H}}_2$ is the bare Hamiltonian or a similarity transformed Hamiltonian truncated up to two-body operators. The braces indicate Kutzelnigg-Mukherjee normal ordering [525, 526], which is used extensively in the definition of the MR-EOM-CC formalism. The various transformations which need to be considered in the ORCA implementation of MR-EOM-CC are summarized in Table 3.53. The table also includes the expressions for the operator components of the various internally contracted cluster operators and the residual equations that must be solved for the various amplitudes. Note that the residual equations are typically of the many-body type (i.e. obtained by setting the corresponding elements of the similarity transformed Hamiltonian to zero). The only exception is the residual equation which defines the t_i^a amplitudes, which is a projected equation of the form

$$R_i^a = \sum_{\mathfrak{m}} w_{\mathfrak{m}} \langle \Phi_{\mathfrak{m}} | \widehat{E}_a^i \widehat{H} | \Phi_{\mathfrak{m}} \rangle.$$

Here, $|\Phi_{\mathfrak{m}}\rangle$ is the $\mathfrak{m}^{\text{th}}$ state included in the state averaged CAS, with weight $w_{\mathfrak{m}}$. The reason the equation for the singles is of the projected form is that it satisfies the Brillouin theorem (i.e. the first order singles vanish for all i and a), whereas the corresponding many-body equation ($\bar{h}_i^a = 0$) does not.

Table 3.54: Details of the three MR-EOM-CC approaches implemented in ORCA

Method	Input Keyword	Operators	Diagonalization Manifold
MR-EOM-T|T†-h-v	`MR-EOM-T|Td`	$\widehat{T}; \widehat{T}^\dagger$	CAS, 2h1p, 1h1p, 2h, 1h, 1p
MR-EOM-T|T†|SXD-h-v	`MR-EOM-T|Td|SXD`	$\widehat{T}; \widehat{T}^\dagger; \widehat{S} + \widehat{X} + \widehat{D}$	CAS, 2h, 1h, 1p
MR-EOM-T|T†|SXD|U-h-v	`MR-EOM`	$\widehat{T}; \widehat{T}^\dagger; \widehat{S} + \widehat{X} + \widehat{D}; \widehat{U}$	CAS, 1h, 1p

Note that there are three different MR-EOM-CC approaches which have been implemented in ORCA. Namely, the current implementation allows for MR-EOM-T|T†-h-v, MR-EOM-T|T†|SXD-h-v and MR-EOM-T|T†|SXD|U-h-v calculations. At this point it is useful to discuss the naming convention used for these approaches. We use a vertical line to separate each transformation involved in the sequence of transformations defining the given MR-EOM-CC approach. For example, T|T†|SXD indicates that a T transformation, is followed by a T† transformation, which is then followed by an SXD transformation. The h-v indicates that the elements of the transformed Hamiltonian have been hermitized (h) and vertex symmetrized (v) before entering the MRCI diagonalization (see Ref. [521] for more information). Essentially, this means that the full eightfold symmetry of the two-electron integrals (and hermiticity of the one-body elements) have been enforced upon the elements of the transformed Hamiltonian. The details of the three MR-EOM-CC approaches are summarized in Table Table 3.54. This table includes the keyword (in the first line of input) used to initiate the calculation in ORCA, the various operators involved, and the configurations included in the final diagonalization manifold. One can clearly see that the MR-EOM-T|T†|SXD|U-h-v approach is the most cost effective, as it only includes the 1h and 1p configurations, beyond the CAS, in the final diagonalization manifold.

Some `%mdci` keywords which are important for controlling MR-EOM-CC calculations are (i.e. default values are given here):

```
%mdci
  STol              1e-7        #Convergence Tolerance on Residual Equations
  MaxIter           100         #Maximum Number of Iterations
  DoSingularPT      false       #Activate the Singular PT/Projection Procedure
  SingularPTThresh  0.01        #Threshold for the Singular PT/Projection
                                #Procedure
  PrintOrbSelect    false       #Print the Eigenvalues of the Orbital Selection
                                #Densities (and R_core and R_virt values)
                                #and Terminate the Calculation
  CoreThresh        0.0         #Core Orbital Selection Threshold
  VirtualThresh     1.0         #Virtual Orbital Selection Threshold
end
```

As discussed below, the orbital selection scheme is activated by adding `!OrbitalSelection` to the simple input line. Keywords that are specific to the CASSCF and MRCI modules are discussed in sections Section 3.13 and Section 3.19, respectively. We note that in MR-EOM-T|T†-h-v and MR-EOM-T|T†|SXD-h-v calculations, it is possible to override the default excitation classes in the final MRCI diagonalization. This is done by specifying `excitations none` and then explicitly setting the excitation flags within a given multiplicity block. For example, if we wanted to have 1h, 1p, 1h1p, 2h and 2h1p excitations in the final diagonalization manifold, we would specify (i.e. here we have requested 6 singlets and have a CAS(6,4) reference):

```
%mrci
    newblock 1 *
        excitations none
        Flags[is]   1
        Flags[sa]   1
        Flags[ia]   1
        Flags[ijss] 1
        Flags[ijsa] 1
        nroots      6
        refs
            cas(6,4)
        end
    end
end
```

The Steps Required to Run an MR-EOM-CC Calculation

To illustrate the various steps required in a typical MR-EOM-CC calculation, we will consider the calculation of the excitation energies of the neutral Fe atom at the MR-EOM-T|T†|SXD|U-h-v level of theory.

State-Averaged CASSCF Calculation

Evidently, the first step is to determine a suitable state-averaged CASSCF reference for the subsequent MR-EOM-CC calculation. In choosing the state-averaged CAS for an MR-EOM-CC calculation, we typically include a few of the low-lying multiplets that have the same character as the (much larger number of) states that we wish to compute in the final MR-EOM calculation. For the neutral Fe atom, we typically have electronic states which have either $4s^2 3d^6$ character or $4s^1 3d^7$ character. From the NIST atomic spectra database [527, 528], we find that the lowest lying ^{5}D multiplet is of $4s^2 3d^6$ character and the higher lying ^{5}F multiplet is of $4s^1 3d^7$ character. Hence, we can set up a state-averaged CASSCF(8,6) calculation (i.e. 8 electrons in 6 orbitals (4s and 3d)) which includes the ^{5}D and ^{5}F states and choose the weights such that the average occupation of the 4s orbital is 1.5. As discussed in Ref. [522], this is done to avoid a preference toward either of the 4s configurations in the state-averaging. We will run the state-averaged CASSCF calculation, making use of the second order DKH (see *The Douglas-Kroll-Hess Method*) method for the inclusion of relativistic effects in a Def2-TZVPP basis (i.e. the DKH-Def2-TZVPP relativistically recontracted basis, listed in section Section 2.7). The input file for the state-averaged CASSCF(8,6) calculation takes the form:

```
!CASSCF DKH-Def2-TZVPP VeryTightSCF DKH

%casscf
  nel 8
  norb 6
  mult 5
  nroots 12
  weights[0] = 0.7, 0.7, 0.7, 0.7, 0.7, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
end

* xyz 0 5
  Fe 0.000000 0.000000 0.000000
end
```

Here, we have requested 12 quintet states (the lowest lying ^{5}D and ^{5}F multiplets) and we have chosen the weights to be 0.7 for the five ^{5}D states and 0.5 for the seven ^{5}F states, such that the overall occupation of the 4s orbital will be

1.5. Once the calculation has converged, it is important to inspect the results printed in the final macro-iteration of the CASSCF calculation (macro-iteration 8 in this case). In this case, we have:

```
MACRO-ITERATION   8:
   --- Inactive Energy E0 = -1249.82392764 Eh
   --- All densities will be recomputed
CI-ITERATION   0:
 -1271.258899450   0.000000000001 (    0.00)
 -1271.258899450   0.000000000001
 -1271.258899450   0.000000000001
 -1271.258899450   0.000000000001
 -1271.258899450   0.000000000001
 -1271.186288591   0.000000000001
 -1271.186288591   0.000000000001
 -1271.186288591   0.000000000002
 -1271.186288591   0.000000000001
 -1271.186288591   0.000000000001
 -1271.186288591   0.000000000001
 -1271.186288591   0.000000000001
   CI-PROBLEM SOLVED
   DENSITIES MADE
   E(CAS)= -1271.222594021 Eh DE=   -1.591616e-12
   --- Energy gap subspaces: Ext-Act = 0.276   Act-Int = 2.469
   N(occ)=  1.50000 1.30000 1.30000 1.30000 1.30000 1.30000
   ||g|| =     4.669702e-05 Max(G)=    1.104493e-05 Rot=68,1
```

Directly below `CI-ITERATION 0`, the final CAS-CI energies are printed, and one observes that they follow the correct degeneracy pattern (i.e. 5 states with energy `-1271.258899450` and 7 states with energy `-1271.186288591`). Furthermore, the final state-averaged CASSCF energy (`E(CAS)= -1271.222594021`) and occupation numbers (`N(occ)= 1.50000 1.30000 1.30000 1.30000 1.30000 1.30000`) are also printed. As expected, the occupation number of the 4s orbital is indeed 1.5, while the 3d orbitals each have an occupation of 1.3.

Selection of the States to Include in the MR-EOM-CC Calculation

Once a satisfactory CASSCF reference has been obtained, the next step is to determine the number of states to include in the MR-EOM-CC calculation. From the NIST atomic spectra database, one finds that the higher lying states of $4s^2 3d^6$ and $4s^1 3d^7$ character are either singlets, triplets, or quintets. To figure out how many states should be included in each multiplicity block, one can perform an inexpensive CAS-CI calculation. This is done by reading in the orbitals from the previous CASSCF calculation (here they are stored in the file `CAS.gbw`) and requesting a single iteration (i.e. using the `NoIter` keyword) of a state-averaged CASSCF calculation:

```
!CASSCF DKH-Def2-TZVPP ExtremeSCF DKH NoIter

!MOREAD
%moinp "CAS.gbw"

%casscf
  nel 8
  norb 6
  mult 5,3,1
  nroots 15,90,55
end

* xyz 0 5
  Fe 0.000000 0.000000 0.000000
end
```

Here, after some experimentation, we have chosen 15 quintets, 90 triplets and 55 singlets. It is important that we calculate states up to sufficiently high energy (i.e. all the states that are of interest) and it is imperative that we have complete multiplets. Hence, several iterations of this procedure might be required to choose the proper number

of states for each multiplet. The relevant section of the output file which should be analyzed is the `SA-CASSCF TRANSITION ENERGIES`. For the above calculation, we obtain (i.e. only the CAS-CI energies for the first 33 roots are shown here):

```
-----------------------------
SA-CASSCF TRANSITION ENERGIES
-----------------------------

LOWEST ROOT (ROOT 0 ,MULT 5) =  -1271.258899450 Eh -34592.713 eV

STATE   ROOT MULT  DE/a.u.    DE/eV    DE/cm**-1
   1:    1    5   0.000000     0.000      0.0
   2:    2    5   0.000000     0.000      0.0
   3:    3    5   0.000000     0.000      0.0
   4:    4    5   0.000000     0.000      0.0
   5:    5    5   0.072611     1.976  15936.2
   6:    6    5   0.072611     1.976  15936.2
   7:    7    5   0.072611     1.976  15936.2
   8:    8    5   0.072611     1.976  15936.2
   9:    9    5   0.072611     1.976  15936.2
  10:   10    5   0.072611     1.976  15936.2
  11:   11    5   0.072611     1.976  15936.2
  12:    0    3   0.092859     2.527  20380.2
  13:    1    3   0.092859     2.527  20380.2
  14:    2    3   0.092859     2.527  20380.2
  15:    3    3   0.092859     2.527  20380.2
  16:    4    3   0.092859     2.527  20380.2
  17:    5    3   0.092859     2.527  20380.2
  18:    6    3   0.092859     2.527  20380.2
  19:    7    3   0.092859     2.527  20380.2
  20:    8    3   0.092859     2.527  20380.2
  21:    9    3   0.092859     2.527  20380.2
  22:   10    3   0.092859     2.527  20380.2
  23:   11    3   0.101848     2.771  22353.1
  24:   12    3   0.101848     2.771  22353.1
  25:   13    3   0.101848     2.771  22353.1
  26:   14    3   0.101848     2.771  22353.1
  27:   15    3   0.101848     2.771  22353.1
  28:   16    3   0.101848     2.771  22353.1
  29:   17    3   0.101848     2.771  22353.1
  30:   18    3   0.102559     2.791  22509.1
  31:   19    3   0.102559     2.791  22509.1
  32:   20    3   0.102559     2.791  22509.1
```

Running the MR-EOM-CC Calculation

Now that we have chosen a suitable CASSCF reference and the states that we wish to calculate, we can finally proceed with the MR-EOM calculation. The following input file runs an MR-EOM-T|T†|SXD|U-h-v calculation for 15 quintet, 90 triplet and 55 singlet states of the neutral Fe atom (i.e. the CASSCF orbitals are read from `CAS.gbw`):

```
!MR-EOM  DKH-Def2-TZVPP VeryTightSCF DKH

!MOREAD
%moinp "CAS.gbw"

%method frozencore fc_ewin end

%casscf
  nel 8
  norb 6
```

(continues on next page)

(continued from previous page)

```
  mult 5
  nroots 12
  weights[0] = 0.7, 0.7, 0.7, 0.7, 0.7, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
end

%mdci
  ewin -6, 10000
  STol 1e-7
end

%mrci
  ewin -6, 10000
  MaxIter 200
  newblock 5 *
    nroots 15
    refs cas(8,6) end
  end
  newblock 3 *
    nroots 90
    refs cas(8,6) end
  end
  newblock 1 *
    nroots 55
    refs cas(8,6) end
  end
end

* xyz 0 5
  Fe 0.000000 0.000000 0.000000
end
```

Note that since the default frozen core settings exclude the 3p orbitals from the correlation treatment, we have used an energy window (i.e. the line `ewin -6, 10000` in both the `%mdci` and `%mrci` blocks) such that they are included in the current calculation. We note that a detailed discussion of the input and output of an MR-EOM-CC calculation has already been given in section Section 3.20 and thus, we do not repeat it here. It is important to reiterate that one should always inspect the values of the largest (T, S and U) amplitudes. Ideally, the largest amplitudes should be smaller than 0.1 and should not exceed 0.15. If some amplitudes are larger than 0.15, it might be necessary to revisit the definition of the CAS and the weights used. For the T amplitudes, an alternative solution is to use the projection/singular PT scheme discussed in section Section 3.20.4 below.

As discussed in section Section 3.20, the excitation energies are printed under the heading `TRANSITION ENERGIES`. For the current calculation, we obtain the following results (only the results for 33 states are shown here):

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is  -1271.833871822 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    5    -1    0    0      0.000      0.000       0.0
  1    5    -1    1    0      0.000      0.000       0.0
  2    5    -1    2    0      0.000      0.000       0.0
  3    5    -1    3    0      0.000      0.000       0.0
  4    5    -1    4    0      0.000      0.000       0.0
  5    5    -1    5    0     33.901      0.922    7440.4
  6    5    -1    6    0     33.901      0.922    7440.4
  7    5    -1    7    0     33.901      0.922    7440.4
  8    5    -1    8    0     33.901      0.922    7440.4
  9    5    -1    9    0     33.901      0.922    7440.4
```

(continues on next page)

(continued from previous page)

```
 10    5    -1   10    0     33.901      0.922    7440.4
 11    5    -1   11    0     33.901      0.922    7440.4
 12    3    -1    0    1     54.743      1.490   12014.8
 13    3    -1    1    1     54.743      1.490   12014.8
 14    3    -1    2    1     54.743      1.490   12014.8
 15    3    -1    3    1     54.743      1.490   12014.8
 16    3    -1    4    1     54.743      1.490   12014.8
 17    3    -1    5    1     54.743      1.490   12014.8
 18    3    -1    6    1     54.743      1.490   12014.8
 19    5    -1   12    0     78.790      2.144   17292.5
 20    5    -1   13    0     78.790      2.144   17292.5
 21    5    -1   14    0     78.790      2.144   17292.5
 22    3    -1    7    1     95.413      2.596   20940.8
 23    3    -1    8    1     95.413      2.596   20940.8
 24    3    -1    9    1     95.413      2.596   20940.8
 25    3    -1   10    1     95.413      2.596   20940.8
 26    3    -1   11    1     95.413      2.596   20940.8
 27    3    -1   12    1     95.413      2.596   20940.8
 28    3    -1   13    1     95.413      2.596   20940.8
 29    3    -1   14    1     95.413      2.596   20940.8
 30    3    -1   15    1     95.413      2.596   20940.8
 31    3    -1   16    1     95.413      2.596   20940.8
 32    3    -1   17    1     95.413      2.596   20940.8
```

It is also important to recall that one should always inspect the reference weights for each state, as only states which are dominated by reference space configurations can be treated accurately at the MR-EOM level of theory. Generally, the reference weights should be larger than (or close to) 0.9. In each multiplicity block, the individual state energies and reference weights can be found following convergence of the MRCI procedure, under the heading `CI-RESULTS` (see section Section 3.20 for a more detailed discussion).

Approximate Inclusion of Spin-Orbit Coupling Effects in MR-EOM-CC Calculations

The effects of spin-orbit coupling can approximately be included in MR-EOM-CC calculations using the SOC submodule of the MRCI module. This can be viewed as a first order approximation to the inclusion of spin-orbit coupling effects in MR-EOM-CC. In a more rigorous formulation, one would have to consider the various similarity transformations of the spin-orbit coupling operator. The details of the SOC submodule of the MRCI module have already been discussed in detail in Sec. Section 3.19.14 and its usage within the MR-EOM formalism is identical to that discussed therein. Let us consider the calculation of spin-orbit coupling effects in the excitation spectrum of the neutral Fe atom considered in the previous section. The input file for this calculation is:

```
!MR-EOM DKH-Def2-TZVPP ExtremeSCF DKH

%method frozencore fc_ewin end

%casscf
  nel 8
  norb 6
  mult 5
  nroots 12
  weights[0] = 0.7, 0.7, 0.7, 0.7, 0.7, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
  etol 1e-12
  gtol 1e-12
end

%mdci
  ewin -6, 10000
  MaxIter 300
  STol 1e-12
end
```

(continues on next page)

(continued from previous page)

```
%mrci
  ewin -6, 10000
  MaxIter 200
  newblock 5 *
    nroots 15
    refs cas(8,6) end
  end
  newblock 3 *
    nroots 90
    refs cas(8,6) end
  end
  newblock 1 *
    nroots 55
    refs cas(8,6) end
  end
  soc
    dosoc true  #include spin-orbit coupling effects
  end
end

* xyz 0 5
  Fe 0.000000 0.000000 0.000000
end
```

In contrast with the calculation performed in section Section 3.20.1, the convergence thresholds have been tightened in all aspects of the calculation (i.e. the use of the `ExtremeSCF` keyword, `etol` and `gtol` (CASSCF energy and orbital gradient convergence tolerance) are set to 1×10^{-12} and the convergence tolerance for the residuals in the MR-EOM-CC amplitude iterations have been set to 1×10^{-12}). We note that with the use of the `ExtremeSCF` keyword, the convergence tolerance on the energy (`Etol`) and residual (`Rtol`) in the MRCI portion of the calculation are also set to 1×10^{-12}. Although it is not absolutely necessary, we have used very strict convergence thresholds to preserve the degeneracies of the various multiplets as much as possible. The output of spin-orbit corrected MR-EOM-CC spectrum appears under the heading `SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION DIPOLE MOMENTS`:

```
-----------------------------------------------------------------------------------
↪---------------------
      SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪---------------------
      Transition         Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                          (eV)      (cm-1)    (nm)   (*population)  (au**2)    (au)␣
↪     (au)      (au)
-----------------------------------------------------------------------------------
↪---------------------
```

It is possible to obtain more accurate results by performing an MR-EOM-T|T†|SXD-h-v calculation and including the 1h1p excitations. It is important to note that these calculations are significantly more expensive. As discussed above, to run an MR-EOM-T|T†|SXD-h-v calculation, the keyword `MR-EOM-T|Td|SXD` must appear in the first line of input and, in order to activate the 1h1p excitations in each multiplicity block of the MRCI calculation, the `%mrci` block takes the form:

```
%mrci
  ewin -6, 10000
  MaxIter 200
  newblock 5 *
    nroots 15
    excitations none
    Flags[is] 1
```

(continues on next page)

(continued from previous page)

```
    Flags[sa] 1
    Flags[ia] 1
    Flags[ijss] 1
    refs cas(8,6) end
  end
  newblock 3 *
    nroots 90
    excitations none
    Flags[is] 1
    Flags[sa] 1
    Flags[ia] 1
    Flags[ijss] 1
    refs cas(8,6) end
  end
  newblock 1 *
    nroots 55
    excitations none
    Flags[is] 1
    Flags[sa] 1
    Flags[ia] 1
    Flags[ijss] 1
    refs cas(8,6) end
  end
  soc
    dosoc true
  end
end
```

We use `excitations none` to set the default excitation flags to false and then manually set the 1h (`Flags[is]`), 1p (`Flags[sa]`), 1h1p (`Flags[ia]`) and 2h (`Flags[ijss]`) excitation flags to true.

Warning

- Currently, MR-EOM-T|T†|SXD|U-h-v calculations can only be run with the default excitation classes in the final MRCI (i.e. 1h and 1p). Any other input options for the excitation flags will automatically be overwritten and set to the default values.
- Only the inclusion of spin-orbit coupling effects has been tested for MR-EOM-CC calculations. Other features that are available in the MRCI module (e.g. spin-spin coupling, magnetic property calculations, etc.) have not been tested for use within MR-EOM calculations.

A Projection/Singular PT Scheme to Overcome Convergence Issues in the T Amplitude Iterations

In certain cases, there may be nearly singular T_2 amplitudes (often, but not always large in magnitude), which can cause convergence issues in the solution of the T amplitude equations. Hence, it is sometimes necessary to discard some of the amplitudes to remedy these convergence problems. The nearly singular T_2 amplitudes are of the form t^{ab}_{wx}, where (w, x) is a pair of active orbitals which corresponds to a small eigenvalue (pair occupation number n_{wx}) of the two-body reduced density matrix (RDM). When nearly singular amplitudes are present, it is possible to employ a singular PT/projection scheme (i.e. Scheme I described in Ref. [516]), using the two-body RDM as the metric matrix, to discard these nearly singular amplitudes and replace them with suitable perturbative estimates. As a first example, let us consider the following calculation on the cyclopentadiene molecule:

```
!MR-EOM def2-SVP VeryTightSCF

%casscf
  nel 4
```

(continues on next page)

(continued from previous page)

```
  norb 4
  nroots 2
  mult 3
end

%mdci
  STol 1e-7
  MaxIter 60
end

%mrci
  newblock 1 *
    nroots 3
    refs cas(4,4) end
  end
  newblock 3 *
    nroots 3
    refs cas(4,4) end
  end
end

* xyz 0 3
  H       -0.879859        0.000000        1.874608
  H        0.879859        0.000000        1.874608
  H        0.000000        2.211693        0.612518
  H        0.000000       -2.211693        0.612518
  H        0.000000        1.349811       -1.886050
  H        0.000000       -1.349811       -1.886050
  C        0.000000        0.000000        1.215652
  C        0.000000       -1.177731        0.285415
  C        0.000000        1.177731        0.285415
  C        0.000000       -0.732372       -0.993420
  C        0.000000        0.732372       -0.993420
*
```

The T amplitude iterations do not converge after 60 iterations and show no signs of convergence (i.e. final largest residual of 0.000458135 and oscillatory behavior over a significant portion of the iterations). If we inspect the largest T amplitudes,

```
--------------------
LARGEST T AMPLITUDES
--------------------
  19-> 24  19-> 24       0.043103
  19-> 23  19-> 23       0.031162
  11-> 25  11-> 25       0.028458
  19-> 41  19-> 41       0.027950
  11-> 47  11-> 47       0.027026
  19-> 22  19-> 22       0.025163
  19-> 21  19-> 21       0.022167
  15-> 26  15-> 26       0.022084
  11-> 47  11-> 25       0.022033
  11-> 25  11-> 47       0.022033
  19-> 24  19-> 29       0.021769
  19-> 29  19-> 24       0.021769
  19-> 36  19-> 36       0.020986
  17-> 38  17-> 38       0.019743
  19-> 41  16-> 36       0.019107
  18-> 40  18-> 40       0.017949
```

one can see that there are no unusually large amplitudes. If we turn on the singular PT/projection scheme by adding the line `DoSingularPT true` to the `%mdci` block:

```
%mdci
  STol 1e-7
  MaxIter 60
  DoSingularPT true
end
```

and rerun the calculation, we find that the T amplitude iterations now successfully converge in 22 iterations. If we look at the largest T amplitudes:

```
--------------------
LARGEST T AMPLITUDES
--------------------
  11-> 25  11-> 25       0.028440
  11-> 47  11-> 47       0.027027
  15-> 26  15-> 26       0.022069
  11-> 47  11-> 25       0.022031
  11-> 25  11-> 47       0.022031
  19-> 41  19-> 41       0.020463
  17-> 38  17-> 38       0.018288
  11-> 43  11-> 43       0.017250
  11-> 39  11-> 39       0.016838
  15-> 27  15-> 27       0.016001
  13-> 26  13-> 26       0.015985
  16-> 36  16-> 36       0.015759
  19-> 41  16-> 36       0.015697
  18-> 40  18-> 40       0.015376
  17-> 31  17-> 31       0.015074
  18-> 40  17-> 38       0.014470
```

most of the amplitudes corresponding to the active pair $(w, x) = (19, 19)$ no longer appear in the list (i.e. they are nearly singular amplitudes which have been projected out). The only one that does appear in the list, corresponds to a projected perturbative estimate (e.g. `19-> 41 19-> 41 0.020463`).

By default, when the singular PT/projection scheme is active, the amplitudes t_{wx}^{ab} for which the pair occupation numbers satisfy $n_{wx} < 0.01$ (i.e. `SingularPTThresh = 0.01`), are replaced by perturbative amplitudes in the procedure. However, in some cases, it might be necessary to increase the `SingularPTThresh` threshold beyond the default value to achieve convergence. One such example is the ferrocene molecule. Consider the following calculation:

```
!MR-EOM def2-SVP

%casscf
  nel 6
  norb 5
  mult  1,3
  nroots 5,6
end

%mdci
  DoSingularPT true
  MaxIter 50
end

%mrci
  newblock 1 *
    nroots 18
    refs cas(6,5) end
  end
  newblock 3 *
    nroots 10
    refs cas(6,5) end
  end
```

(continues on next page)

(continued from previous page)

```
end

* xyz 0 1
  Fe 0.000000  0.000000  0.000000
  C  0.000000  1.220080  1.650626
  C -1.160365  0.377025  1.650626
  C -0.717145 -0.987065  1.650626
  C  0.717145 -0.987065  1.650626
  C  1.160365  0.377025  1.650626
  C  0.000000  1.220080 -1.650626
  C  1.160365  0.377025 -1.650626
  C  0.717145 -0.987065 -1.650626
  C -0.717145 -0.987065 -1.650626
  C -1.160365  0.377025 -1.650626
  H  0.000000  2.306051  1.635648
  H -2.193184  0.712609  1.635648
  H -1.355463 -1.865634  1.635648
  H  1.355463 -1.865634  1.635648
  H  2.193184  0.712609  1.635648
  H  0.000000  2.306051 -1.635648
  H  2.193184  0.712609 -1.635648
  H  1.355463 -1.865634 -1.635648
  H -1.355463 -1.865634 -1.635648
  H -2.193184  0.712609 -1.635648
end
```

The T amplitude iterations do not converge after 50 iterations, even though the singular PT/projection scheme is activated. If we increase `SingularPTThresh` to 0.05 by adding `SingularPTThresh 0.05` to the `%mdci` block:

```
%mdci
  DoSingularPT true
  SingularPTThresh 0.05
  MaxIter 50
end
```

the T amplitude iterations successfully converge in 25 iterations.

In conclusion, it occasionally happens that the T amplitude iterations do not converge. In these cases, the singular PT/projection scheme can be activated (`DoSingularPT true`) to overcome these convergence difficulties. Sometimes, like in the case of ferrocene, it is necessary to adjust the threshold for the singular PT/projection procedure (`SingularPTThresh`) to achieve convergence. If the procedure still fails with larger values of the threshold, then it might be necessary to revisit the definition of the state-averaged CAS.

3.20.5 An Orbital Selection Scheme for More Efficient Calculations of Excitation Spectra with MR-EOM

As described in Ref. [520], the MR-EOM implementation in ORCA can make use of a sophisticated scheme to discard inactive and virtual orbitals, which are not important for the description of the excited states of interest. The selection of inactive core orbitals is based on the eigenvalues of the core orbital selection density

$$D_{i'j'} = D^t_{i'j'} + \frac{\mathrm{Tr}\left(D^t\right)}{\mathrm{Tr}\left(D^s\right) + \mathrm{Tr}\left(D^u\right)} \left(D^s_{i'j'} + D^u_{i'j'}\right), \tag{3.143}$$

in which

$$D^t_{i'j'} = \sum_{w,a,b} t^{ab(1)}_{i'w} \left(2t^{ab(1)}_{j'w} - t^{ba(1)}_{j'w}\right),$$
$$D^s_{i'j'} = \sum_{k,w,a} \left[s^{aw(1)}_{i'k} \left(2^{aw(1)}_{j'k} - s^{aw(1)}_{kj'}\right) + s^{aw(1)}_{ki'} \left(2s^{aw(1)}_{kj'} - s^{aw(1)}_{j'k}\right)\right],$$
$$D^u_{i'j'} = \sum_{k',w,x} u^{wx(1)}_{i'k'} \left(2u^{wx(1)}_{j'k'} - u^{wx(1)}_{k'j'}\right),$$

are respectively, the contributions from the first order $t^{ab(1)}_{i'w}$, $s^{aw(1)}_{i'k}$ and $u^{wx(1)}_{i'k'}$ amplitudes (i.e. note that all amplitudes have at least one active label). Similarly, the selection of virtual orbitals is based upon the eigenvalues of the virtual orbital selection density

$$\rho_{ab} = \rho^t_{ab} + \frac{\mathrm{Tr}\,(\rho^t)}{\mathrm{Tr}\,(\rho^s)}\rho^s_{ab}, \tag{3.144}$$

in which, the contribution ρ^t, from the first order T_2 amplitudes, is given by

$$\rho^t_{ab} = \sum_{k,w,c} t^{ac(1)}_{wk} \left(2t^{bc(1)}_{wk} - t^{cb(1)}_{wk}\right) + \sum_{i',w,c} t^{ac(1)}_{i'w} \left(2t^{bc(1)}_{i'w} - t^{cb(1)}_{i'w}\right), \tag{3.145}$$

and the contribution ρ^s, from the first order S_2 amplitudes, is given by

$$\rho^s_{ab} = \sum_{i',k,w} s^{aw(1)}_{i'k} \left(2s^{bw(1)}_{i'k} - s^{bw(1)}_{ki'}\right) + \sum_{i',w,x} s^{aw(1)}_{xi'} \left(2s^{bw(1)}_{xi'} - s^{bw(1)}_{i'x}\right). \tag{3.146}$$

Diagonalization of the core orbital selection density $D_{i'j'}$ and the virtual orbital selection density ρ_{ab} then yields two respective sets of eigenvalues $\{\lambda_{i'}\}$ and $\{\lambda_a\}$. We have found it useful to compute the ratios,

$$\mathcal{R}_{\text{core}} = \frac{\sum_{i'=0}^{n^{\text{excluded}}_{\text{core}}} \lambda_{i'}}{\sum_{i'=0}^{n_{\text{core}}} \lambda_{i'}} \times 100\%, \tag{3.147}$$

$$\mathcal{R}_{\text{virt}} = \frac{\sum_{a=0}^{n^{\text{excluded}}_{\text{virt}}} \lambda_a}{\sum_{a=0}^{n_{\text{virt}}} \lambda_a} \times 100\%, \tag{3.148}$$

of the sum of the excluded eigenvalues to the sum over all eigenvalues. The orbital selection in the core and virtual subspaces is then based upon the values of these ratios, as will be discussed below.

The orbital selection procedure is activated by adding the keyword `OrbitalSelection` to the first line of input, e.g.

```
! MR-EOM-CC  def2-TZVPP VeryTightSCF OrbitalSelection
```

There are two threshold parameters `CoreThresh` and `VirtualThresh`, which are used to determine which inactive core and virtual orbitals are to be discarded in the orbital selection procedure, respectively. Namely, all inactive core orbitals for which $\mathcal{R}_{\text{core}} <$ `CoreThresh` (i.e. $\mathcal{R}_{\text{core}}$ as defined in Eq. (3.147)) are discarded and all virtual orbitals satisfying the condition $\mathcal{R}_{\text{virt}} <$ `VirtualThresh` (i.e. $\mathcal{R}_{\text{virt}}$ as defined in Eq. (3.148)) are discarded. The default values of these thresholds are `CoreThresh = 0.0` (no core orbital selection) and `VirtualThresh = 1.0`. However, the values of these parameters can easily be changed by redefining them in the `%mdci` block:

```
%mdci
  CoreThresh 1.0
  VirtualThresh 1.0
end
```

Let us consider the calculation of the previous section (*A Projection/Singular PT Scheme to Overcome Convergence Issues in the T Amplitude Iterations*) on ferrocene, with the orbital selection procedure activated (using the default thresholds):

```
!MR-EOM def2-SVP OrbitalSelection

%casscf
  nel 6
  norb 5
  mult   1,3
  nroots 5,6
end

%mdci
  DoSingularPT true
  SingularPTThresh 0.05
  MaxIter 50
end

%mrci
  newblock 1 *
    nroots 18
    refs cas(6,5) end
  end
  newblock 3 *
    nroots 10
    refs cas(6,5) end
  end
end

* xyz 0 1
  Fe 0.000000  0.000000  0.000000
  C  0.000000  1.220080  1.650626
  C -1.160365  0.377025  1.650626
  C -0.717145 -0.987065  1.650626
  C  0.717145 -0.987065  1.650626
  C  1.160365  0.377025  1.650626
  C  0.000000  1.220080 -1.650626
  C  1.160365  0.377025 -1.650626
  C  0.717145 -0.987065 -1.650626
  C -0.717145 -0.987065 -1.650626
  C -1.160365  0.377025 -1.650626
  H  0.000000  2.306051  1.635648
  H -2.193184  0.712609  1.635648
  H -1.355463 -1.865634  1.635648
  H  1.355463 -1.865634  1.635648
  H  2.193184  0.712609  1.635648
  H  0.000000  2.306051 -1.635648
  H  2.193184  0.712609 -1.635648
  H  1.355463 -1.865634 -1.635648
  H -1.355463 -1.865634 -1.635648
  H -2.193184  0.712609 -1.635648
end
```

The details of the orbital selection procedure are printed under the heading `ORBITAL SELECTION`:

```
-----------------------------------------------
ORBITAL SELECTION
-----------------------------------------------

T1 is NOT used in the construction of the orbital selection densities
Factor (in percent) for inactive (core) orbital selection
↪          ...      0.000000000
Factor (in percent) for virtual orbital selection
↪          ...      1.000000000
Inactive orbitals before selection:    15 ...   44 ( 30 MO's/ 60 electrons)
```

(continues on next page)

(continued from previous page)

```
Virtual orbitals before selection:     50 ...  220 (171 MO's              )
Inactive orbitals after selection:     15 ...   44 ( 30 MO's/ 60 electrons)
Virtual orbitals after selection:      50 ...  126 ( 77 MO's              )

-------------------------------------------
TIMINGS FOR THE ORBITAL SELECTION PROCEDURE
-------------------------------------------

Total Time for Orbital Selection            ...       61.470 sec

First Half Transformation                   ...       58.041 sec ( 94.4%)
Second Half Transformation                  ...        1.789 sec (  2.9%)
Formation of Orbital Selection Densities    ...        1.629 sec (  2.7%)
Core Orbital Selection                      ...        0.000 sec (  0.0%)
Virtual Orbital Selection                   ...        0.003 sec (  0.0%)
Finalization of Orbitals                    ...        0.006 sec (  0.0%)
```

Comparing the number of virtual orbitals before the orbital selection procedure (171) with the number that are left after orbital selection (77), we see that more than half have been discarded (94). The canonical calculation (without orbital selection) takes 149373 seconds to run and yields the following excitation energies:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is  -1648.190045042 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    -1    0    0      0.000      0.000       0.0
  1    3    -1    0    1     65.110      1.772   14289.9
  2    3    -1    1    1     65.110      1.772   14289.9
  3    3    -1    2    1     70.413      1.916   15454.0
  4    3    -1    3    1     70.413      1.916   15454.0
  5    3    -1    4    1     95.979      2.612   21065.0
  6    3    -1    5    1     95.979      2.612   21065.0
  7    1    -1    1    0    105.302      2.865   23111.1
  8    1    -1    2    0    105.302      2.865   23111.1
  9    1    -1    3    0    107.034      2.913   23491.4
 10    1    -1    4    0    107.034      2.913   23491.4
 11    1    -1    5    0    160.595      4.370   35246.6
 12    1    -1    6    0    160.596      4.370   35246.6
 13    3    -1    6    1    164.694      4.482   36146.1
 14    3    -1    7    1    165.379      4.500   36296.6
 15    3    -1    8    1    165.379      4.500   36296.6
 16    3    -1    9    1    171.464      4.666   37632.1
 17    1    -1    7    0    208.587      5.676   45779.6
 18    1    -1    8    0    208.587      5.676   45779.6
 19    1    -1    9    0    213.093      5.799   46768.6
 20    1    -1   10    0    213.093      5.799   46768.6
 21    1    -1   11    0    216.225      5.884   47456.0
 22    1    -1   12    0    220.230      5.993   48334.9
 23    1    -1   13    0    220.230      5.993   48334.9
 24    1    -1   14    0    224.583      6.111   49290.3
 25    1    -1   15    0    224.583      6.111   49290.3
 26    1    -1   16    0    237.914      6.474   52216.0
 27    1    -1   17    0    237.914      6.474   52216.0
```

In contrast, the calculation with the orbital selection procedure activated runs in 28977 seconds (a factor of 5 speedup) and produces the following excitation energies:

```
-------------------
TRANSITION ENERGIES
-------------------

The lowest energy is  -1647.788478559 Eh

State Mult Irrep Root Block    mEh          eV      1/cm
  0    1    -1    0    0      0.000      0.000       0.0
  1    3    -1    0    1     65.112      1.772   14290.4
  2    3    -1    1    1     65.134      1.772   14295.3
  3    3    -1    2    1     70.520      1.919   15477.3
  4    3    -1    3    1     70.520      1.919   15477.3
  5    3    -1    4    1     96.105      2.615   21092.7
  6    3    -1    5    1     96.134      2.616   21099.0
  7    1    -1    1    0    105.415      2.868   23136.0
  8    1    -1    2    0    105.450      2.869   23143.5
  9    1    -1    3    0    107.294      2.920   23548.3
 10    1    -1    4    0    107.294      2.920   23548.3
 11    1    -1    5    0    161.082      4.383   35353.4
 12    1    -1    6    0    161.094      4.384   35356.0
 13    3    -1    6    1    164.786      4.484   36166.4
 14    3    -1    7    1    165.465      4.503   36315.4
 15    3    -1    8    1    165.465      4.503   36315.5
 16    3    -1    9    1    171.542      4.668   37649.1
 17    1    -1    7    0    208.853      5.683   45838.0
 18    1    -1    8    0    208.853      5.683   45838.0
 19    1    -1    9    0    213.419      5.807   46840.1
 20    1    -1   10    0    213.419      5.807   46840.1
 21    1    -1   11    0    216.526      5.892   47521.9
 22    1    -1   12    0    220.611      6.003   48418.4
 23    1    -1   13    0    220.611      6.003   48418.5
 24    1    -1   14    0    225.135      6.126   49411.5
 25    1    -1   15    0    225.136      6.126   49411.5
 26    1    -1   16    0    238.388      6.487   52320.1
 27    1    -1   17    0    238.388      6.487   52320.1
```

We note that the excitation energies in the orbital selection procedure agree very nicely with those of the canonical calculation. However, the total energies are significantly different, as we currently have not implemented a procedure to correct them. Hence, the following warning is particularly important.

WARNING

- The orbital selection procedure should only be used for the calculation of excitation energies. Total energies computed with the orbital selection procedure have not been corrected and can differ greatly from the canonical results.

Before leaving the discussion of the orbital selection procedure, we note that there is also a keyword `PrintOrbSelect`, which can be added to the `%mdci` block to print the eigenvalues of the inactive core orbital selection and virtual orbital selection densities and the corresponding values of $\mathcal{R}_{\text{core}}$ and $\mathcal{R}_{\text{virt}}$ defined in Eqs. (3.147) and (3.148), respectively. This is useful if one wants to manually select the orbitals to discard in the orbital selection procedure by adjusting the values of `CoreThresh` and `VirtualThresh`. We note that the program terminates after printing. In the case of the calculation on ferrocene, if we modify the `%mdci` block to read

```
%mdci
  DoSingularPT true
  SingularPTThresh 0.05
  MaxIter 50
  PrintOrbSelect True
end
```

we find the following information in the `ORBITAL SELECTION` section of the output (only the first 50 values for the virtual orbital selection density are shown here):

```
Eigenvalues and corresponding R_core values for the core orbital selection density

Orbital         Eigenvalue        R_core
   0            0.00026936        0.419318
   1            0.00027080        0.840883
   2            0.00038739        1.443947
   3            0.00038739        2.047011
   4            0.00040299        2.674355
   5            0.00040299        3.301700
   6            0.00077636        4.510285
   7            0.00086085        5.850394
   8            0.00086085        7.190503
   9            0.00091850        8.620358
  10            0.00091850       10.050213
  11            0.00112826       11.806598
  12            0.00115561       13.605563
  13            0.00137961       15.753236
  14            0.00137961       17.900908
  15            0.00139093       20.066210
  16            0.00139093       22.231512
  17            0.00143349       24.463072
  18            0.00143350       26.694633
  19            0.00148539       29.006985
  20            0.00148539       31.319338
  21            0.00173415       34.018940
  22            0.00224131       37.508054
  23            0.00224132       40.997171
  24            0.00533017       49.294785
  25            0.00533019       57.592429
  26            0.00658679       67.846267
  27            0.00662033       78.152314
  28            0.00701718       89.076149
  29            0.00701719      100.000000

 Eigenvalues and corresponding R_virt values for the virtual orbital selection␣
↪density

Orbital         Eigenvalue        R_virt
   0            0.00000119        0.000450
   1            0.00000119        0.000899
   2            0.00000134        0.001404
   3            0.00000134        0.001909
   4            0.00000136        0.002423
   5            0.00000177        0.003091
   6            0.00000178        0.003764
   7            0.00000178        0.004437
   8            0.00000215        0.005248
   9            0.00000224        0.006096
  10            0.00000224        0.006944
  11            0.00000238        0.007844
  12            0.00000347        0.009154
  13            0.00000347        0.010465
  14            0.00000364        0.011841
  15            0.00000386        0.013299
  16            0.00000396        0.014793
  17            0.00000396        0.016287
  18            0.00000437        0.017937
  19            0.00000437        0.019587
  20            0.00000499        0.021472
  21            0.00000499        0.023357
  22            0.00000794        0.026354
  23            0.00000794        0.029352
```

(continues on next page)

(continued from previous page)

```
  24            0.00000819        0.032447
  25            0.00000819        0.035543
  26            0.00000927        0.039044
  27            0.00000927        0.042546
  28            0.00001002        0.046332
  29            0.00001002        0.050119
  30            0.00001137        0.054415
  31            0.00001137        0.058711
  32            0.00001158        0.063086
  33            0.00001158        0.067461
  34            0.00001381        0.072678
  35            0.00001381        0.077894
  36            0.00001417        0.083249
  37            0.00001417        0.088604
  38            0.00001465        0.094137
  39            0.00001495        0.099785
  40            0.00001495        0.105432
  41            0.00001554        0.111302
  42            0.00001554        0.117172
  43            0.00001623        0.123303
  44            0.00001689        0.129685
  45            0.00001754        0.136310
  46            0.00001754        0.142934
  47            0.00001805        0.149752
  48            0.00001805        0.156570
  49            0.00002111        0.164546

.
.
.
```

In conclusion, the orbital selection scheme provides a more efficient way to calculate accurate excitation spectra within the framework of MR-EOM-CC It can be used to extend the applicability of this approach to larger systems and we expect it to be much more effective in larger systems where the chromophore is localized to a small part of the molecule. We reiterate that it is currently limited to the calculation of excitation energies and should not be used if one is interested in total energies.

3.20.6 Nearly Size Consistent Results with MR-EOM-CC by Employing an MR-CEPA(0) Shift in the Final Diagonalization Procedure

One drawback of the MR-EOM-CC methodology is that it is not size-extensive (or size-consistent). The size-extensivity errors arise due to the final uncontracted MR-CI diagonalization step. Namely, they result from the components of the eigenvectors of the transformed Hamiltonian, which lie outside of the CASSCF reference space (e.g. 1h, 1p, etc. configurations). As more of the excitation classes are included through the successive similarity transformations of the Hamiltonian, the size of the final diagonalization manifold is greatly decreased resulting in much smaller size-extensivity errors upon going from MR-EOM-T|T†-h-v to MR-EOM-T|T†|SXD|U-h-v. To illustrate this, let us consider the O_2—O_2 dimer where the O_2 molecules are separated by a large distance. For the O_2 monomer, we employ a minimal active space consisting of 2 electrons distributed amongst the two π^* orbitals and we only consider the ground $^3\Sigma_g^-$ state (no state-averaging). In the MR-EOM-CC calculations, we also calculate the higher lying $^1\Delta_g$ and $^1\Sigma_g^+$ singlet states. For example, the input file for the MR-EOM-T|T†|SXD|U-h-v calculation is given by:

```
!MR-EOM AUG-CC-PVTZ EXTREMESCF

%casscf
  nel 2
  norb 2
  nroots 1
```

(continues on next page)

(continued from previous page)

```
  mult 3
end

%mdci
  MaxIter 300
  STol 1e-12
end

%mrci
  newblock 1 *
    nroots 3
    refs cas(2,2) end
  end
  newblock 3 *
    nroots 1
    refs cas(2,2) end
  end
end

* xyz 0 3
  O  0.00000000    -0.00000000    -0.60500000
  O -0.00000000     0.00000000     0.60500000
*
```

In the case of the dimer, we take the reference state as the coupled quintet state which is formed as the product ${}^3\Sigma_g^+ \otimes {}^3\Sigma_g^+$ of the monomer states. We note that at large separation, in the non-interacting limit, the dimer state energies can be decomposed as the sum of monomer state energies. There are various possibilities, taking into account the degeneracies of the various states:

1. a singlet, a triplet, and a quintet with energy $E({}^3\Sigma_g^- + {}^3\Sigma_g^-)$,
2. four triplets with energy $E({}^3\Sigma_g^- + {}^1\Delta_g)$,
3. two triplets with energy $E({}^3\Sigma_g^- + {}^1\Sigma_g^+)$,
4. four singlets with energy $E({}^1\Delta_g + {}^1\Delta_g)$,
5. four singlets with energy $E({}^1\Delta_g + {}^1\Sigma_g^+)$,
6. a singlet with energy $E({}^1\Sigma_g^+ + {}^1\Sigma_g^+)$.

Hence, in the final diagonalization step of the MR-EOM-CC calculation, we must ask for 10 singlets, 7 triplets and 1 quintet. The input file for the MR-EOM-T|T†|SXD|U-h-v calculation on the dimer is given by:

```
!MR-EOM AUG-CC-PVTZ EXTREMESCF

%casscf
  nel 4
  norb 4
  nroots 1
  mult 5
  etol 1e-13
  gtol 1e-13
end

%mdci
  MaxIter 300
  STol 1e-12
end

%mrci
  newblock 1 *
    nroots 10
```

(continues on next page)

(continued from previous page)

```
    refs cas(4,4) end
  end
  newblock 3 *
    nroots 7
    refs cas(4,4) end
  end
  newblock 5 *
    nroots 1
    refs cas(4,4) end
  end
end

* xyz 0 5
  O                 0.00000000     0.00000000  -500.60500000
  O                 0.00000000    -0.00000000  -499.39500000
  O                -0.60500000     0.00000000   500.00000000
  O                 0.60500000    -0.00000000   500.00000000
*
```

In Table Table 3.55, we have compiled the results of the size consistency test, taking the difference of the dimer state energies (at large separation) and the sum of the monomer state energies (in mE_{h}). It is evident that as more excitation classes are included in the similarity transformed Hamiltonian and the size of the final diagonalization manifold is decreased, the size-consistency errors decrease. Of particular note are the results for the MR-EOM-T|T†|SXD|U-h-v approach (only includes 1h and 1p configurations in the final diagonalization manifold), for which the largest deviation is 1.25×10^{-2} mE_{h}. The much larger deviations for the MR-EOM-T|T†|SXD-h-v approach clearly demonstrate the large effect that the 2h excitations have on the size-consistency errors.

Table 3.55: Test for size consistency in MR-EOM-CC Differences in energy (in mE$_{\mathrm{h}}$) between the $_2$—O$_2$ dimer energies (at large separation) and the sum of the monomer energies for the ground state and various excited states. The results were obtained in an aug-cc-pVTZ basis using minimal active spaces.

	T|T†-h-v	T|T†|SXD-h-v (with 1h1p)	T|T†|SXD-h-v	T|T†|SXD|U-h-v
$\Delta E(^3\Sigma_g^- + {}^3\Sigma_g^-)$	12.74	2.77	1.11	1.00×10^{-5}
$\Delta E(^3\Sigma_g^- + {}^1\Delta_g)$	14.20	3.84	1.85	1.54×10^{-4}
$\Delta E(^3\Sigma_g^- + {}^1\Sigma_g^+)$	17.21	5.52	2.83	4.13×10^{-4}
$\Delta E(^1\Delta_g + {}^1\Delta_g)$	15.69	3.10	2.31	5.26×10^{-3}
$\Delta E(^1\Delta_g + {}^1\Sigma_g^+)$	18.83	7.52	4.76	5.89×10^{-3}
$\Delta E(^1\Sigma_g^+ + {}^1\Sigma_g^+)$	22.34	10.75	7.31	1.25×10^{-2}

To reduce the size-consistency errors, one can make use of the MR-CEPA(0) shift in the final diagonalization step. This MR-CEPA(0) shift can easily be activated by adding the line

```
citype mrcepa_0
```

to the beginning of the `%mrci` block. The results of the size-consistency test with the use of the MR-CEPA(0) shift are tabulated in Table Table 3.56. For each of the methods, we see a marked improvement over the results of Table *Test for size consistency in MR-EOM-CC Differences in energy (in mE_\text{h}) between the _2—O_2 dimer energies (at large separation) and the sum of the monomer energies for the ground state and various excited states. The results were obtained in an aug-cc-pVTZ basis using minimal active spaces.*, which do not make use of the MR-CEPA(0) shift. The greatest improvement occurs in the MR-EOM-T|T†|SXD-h-v and the MR-EOM-T|T†|SXD|U-h-v results. Namely, the errors in the former case are on the order of nano Hartrees, while the errors in the MR-EOM-T|T†|SXD|U-h-v results are not detectable (sub-nano Hartree), as the energy is only printed with nine decimal places. It is interesting to note that upon adding the 1h1p configurations to the diagonalization manifold in the MR-EOM-T|T†|SXD-h-v calculations (i.e. with 1h1p), the size-consistency errors increase greatly. Hence, it appears that the use of the MR-CEPA(0) shift is most effective at reducing the size-consistency errors resulting from the presence of the 1h, 1p and 2h configurations in the final diagonalization manifold. In any case, one can easily take advantage of this approach to obtain nearly size-consistent results with both the MR-EOM-T|T†|SXD-h-v and MR-EOM-T|T†|SXD|U-

h-v methods.

Table 3.56: Test for size consistency in MR-EOM-CC using the MR-CEPA(0) shift: Differences in energy (in mE_h) between the O_2—O_2 dimer energies (at large separation) and the sum of the monomer energies for the ground state and various excited states. The results were obtained in an aug-cc-pVTZ basis using minimal active spaces and the MR-CEPA(0) shift was applied in the final diagonalization in each case.

	T|T†-h-v	T|T†|SXD-h-v (with 1h1p)	T|T†|SXD-h-v	T|T†|SXD|U-h-v
$\Delta E(^3\Sigma_g^- + {}^3\Sigma_g^-)$	2.75×10^{-3}	0.01	2.00×10^{-6}	0.00
$\Delta E(^3\Sigma_g^- + {}^1\Delta_g)$	0.06	0.07	0.00	0.00
$\Delta E(^3\Sigma_g^- + {}^1\Sigma_g^+)$	0.14	0.15	4.00×10^{-6}	0.00
$\Delta E(^1\Delta_g + {}^1\Delta_g)$	0.21	0.22	1.00×10^{-6}	0.00
$\Delta E(^1\Delta_g + {}^1\Sigma_g^+)$	0.42	0.44	5.00×10^{-6}	0.00
$\Delta E(^1\Sigma_g^+ + {}^1\Sigma_g^+)$	0.82	0.87	9.00×10^{-6}	0.00

3.20.7 Perturbative MR-EOM-CCPT

The MR-EOM-T approach was developed for situations where the full accuracy of the iterative MR-EOM-CC method is not required. It performs on par with other multireference perturbation theories such as fic-NEVPT2 and does not have the convergence difficulties with the $\hat{T}$ and $\hat{S}, \hat{X}, \hat{D}$ amplitudes like its iterative parent method as these amplitudes are computed in a non-iterative fashion. The only iterative part of the MR-EOM-T method is the calculation of the $\hat{U}$ amplitudes since they are quick to converge anyways [524].

The setup procedure for the MR-EOM-T method is the same as for the MR-EOM-CC method, and the foregoing also applies to the perturbative variant. Please note that the orbital selection scheme has not been tested with this variant and should be unnecessary anyways since calculations are much faster than with the iterative MR-EOM-CC method.

To invoke the new variant, set up the calculation as you would for an MR-EOM-CC calculation and then add the keyword `DoMREOM_MRPT True` to the `%mdci` block.

The results are interpreted just like results for the iterative MR-EOM-CC method. After transforming the Hamiltonian with the perturbatively estimated amplitudes and the final MRCIS diagonalization step, the final state energies are printed along with their reference weights. For reliable results, we again recommend that the reference weight be >90%.

3.21 Dynamic Correlation Dressed CAS

Non-degenerate multireference perturbation theory (MRPT) methods, such as NEVPT2 or CASPT2, have the 0th order part of the wave function fixed by a preceding CASSCF calculation. The latter can be a problem if the CASSCF states are biased towards a wrong state composition in terms of electron configurations. In these instances, a quasi-degenerate or multi-state formulation is necessary, for example the QD-NEVPT2 described in Section *Quasi-Degenerate SC-NEVPT2*. A drawback of these approaches is that the results depend on the number of included states. The 2nd order Dynamic Correlation Dressed Complete Active Space method (DCD-CAS(2)) is a post-CASSCF MRPT method of the perturb-then-diagonalize kind, i.e. it can modify the CAS wavefunction compared to the previous CASSCF.[529] DCD-CAS(2) offers an alternative uncontracted approach, where a dressed CASCI matrix is constructed. Its diagonalization yields correlated energies and 0th order states that are remixed in the CASCI space under the effect of dynamic correlation.[529] In cases where CASSCF already provides a good 0th order wavefunction, DCD-CAS(2) energies are comparable to NEVPT2.

3.21.1 Theory of Nonrelativistic DCD-CAS(2)

The DCD-CAS(2) method is based on solving the eigenvalue problem of an effective Hamiltonian of the form

$$H_{IJ}^{\text{DCD},S} = \langle \Phi_I^{SS} | H | \Phi_J^{SS} \rangle - \sum_{K \in \text{FOIS}} \frac{\langle \Phi_I^{SS} | H | \tilde{\Phi}_K^{SS} \rangle \langle \tilde{\Phi}_K^{SS} | H | \Phi_J^{SS} \rangle}{E_K^S - E_0^S}$$

for each total spin S separately. The 0th order energies E_K^S of the perturbers $|\tilde{\Phi}_K^{SS}\rangle$ are obtained by diagonalizing the Dyall Hamiltonian in the first-order interacting space (FOIS, the space consisting of all single and double excitations with respect to the CASCI space). The effective Hamiltonian has the form of a CASCI Hamiltonian that is dressed with the effect of dynamic correlation (dynamic correlation dressed, DCD), hence the name for the method. E_0^S is chosen to be the ground state CASSCF energy for the respective total spin S. Since this choice is worse for excited states than for the ground state, excitation energies suffer from a "ground state bias".

For the contribution coming from perturbers in which electrons are excited from two inactive (ij) to two virtual (ab) orbitals, we use (when writing the DCD Hamiltonian in a basis of CASCI states) the alternative expression

$$\langle \Psi_I^{SS} | H^{\text{DCD}}(ij \to ab) | \Psi_J^{SS} \rangle = -\delta_{IJ} E_{\text{MP2}}$$

$$E_{\text{MP2}} = \sum_{ijab} \frac{(ib|ja)^2 - (ib|ja)(ia|jb) + (ia|jb)^2}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j}$$

Since in this version the $ij \to ab$ perturber class does not contribute at all to excitation energies (like it is assumed in the difference-dedicated configuration interaction method), we call this the difference-dedicated DCD-CAS(2) method. Since the $ij \to ab$ class contributes the largest part of the dynamic correlation energy, this also removes the largest part of the ground state bias. This option is used as default in DCD-CAS(2) calculations. In order to also remove the ground state bias from the other perturber classes, we furthermore apply a perturbative correction to the final energies. At first order (which is chosen as default), it takes the form

$$\Delta E_I = -\Delta_I \sum_{K \in \text{FOIS}} \frac{\langle \tilde{\Psi}_I | H | \tilde{\Phi}_K \rangle \langle \tilde{\Phi}_K | H | \tilde{\Psi}_I \rangle}{(E_K - E_0)^2}$$

$$\Delta_I = \langle \tilde{\Psi}_I | H | \tilde{\Psi}_I \rangle - E_0$$

for the correction ΔE_I to the total energy of the Ith DCD-CAS(2) root $|\tilde{\Psi}_I\rangle$.

3.21.2 Treatment of spin-dependent effects

The theory so far is valid for a nonrelativistic or scalar-relativistic Hamiltonian H. If we modify it to a Hamiltonian $H + V$, where V contains effects that are possibly spin-dependent such as spin-orbit coupling (SOC) and spin-spin coupling (SSC), this leads us to a theory [432] which has a similar form as QDPT with all CAS roots included. The form of the spin-dependent DCD-CAS(2) effective Hamiltonian is

$$\langle \Phi_I^{SM} | H^{\text{DCD}} | \Phi_J^{S'M'} \rangle = \delta_{SS'} \delta_{MM'} H_{IJ}^{\text{DCD},S,\text{corr}} + \langle \Phi_I^{SM} | V | \Phi_J^{S'M'} \rangle.$$

$$\mathbf{H}^{\text{DCD},S,\text{corr}} = \mathbf{C}^{\text{DCD}} \mathbf{E} (\mathbf{C}^{\text{DCD}})^T.$$

In order to construct it, we first need to solve the scalar-relativistic DCD-CAS(2) problem to construct the matrix $\mathbf{H}^{\text{DCD},S,\text{corr}}$ from the bias corrected energies $\mathbf{E}$ and DCD-CAS(2) CI coefficients $\mathbf{C}^{\text{DCD}}$ and then calculate the matrix elements of the operators contributing to V in the basis of CSFs $|\Phi_I^{SM}\rangle$.

Zero field splitting D tensors are extracted using the effective Hamiltonian technique, i.e. fitting the model Hamiltonian to a des Cloizeaux effective Hamiltonian that is constructed from the relativistic states and energies by projection onto the nonrelativistic multiplet (see section *Zero-Field Splitting* and the reference [530]). There are limitations to this approach if spin orbit coupling becomes so strong that the relativistic states cannot uniquely be assigned to a single nonrelativistic spin multiplet.

Hyperfine A-matrices and Zeeman g-matrices for individual Kramers doublets consisting of states $|\Phi\rangle, |\bar{\Phi}\rangle$ are extracted by comparing the spin Hamiltonians

$$H_{\text{Zeeman}} = \mu_B \vec{B} \cdot g \cdot \vec{S}$$

$$H_{\text{HFC}} = \sum_A \vec{I}^A \cdot A^A \cdot \vec{S}$$

to the matrix representation of the many-electron Zeeman and HFC operators in the basis of the Kramers doublet. This yields [432]

$$\begin{aligned} g_{k1} &= 2\Re\langle\bar{\Phi}|L_k + g_e S_k|\Phi\rangle \\ g_{k2} &= 2\Im\langle\bar{\Phi}|L_k + g_e S_k|\Phi\rangle \\ g_{k3} &= 2\langle\Phi|L_k + g_e S_k|\Phi\rangle \end{aligned}$$

$$\begin{aligned} A_{k1} &= -2\gamma_A\Re\langle\bar{\Phi}|B_k^{\text{HFC}}(\vec{R}_A)|\Phi\rangle \\ A_{k2} &= -2\gamma_A\Im\langle\bar{\Phi}|B_k^{\text{HFC}}(\vec{R}_A)|\Phi\rangle \\ A_{k3} &= -2\gamma_A\langle\Phi|B_k^{\text{HFC}}(\vec{R}_A)|\Phi\rangle \end{aligned}$$

where $B_k^{\text{HFC}}(\vec{R}_A)$ is the kth component of the magnetic hyperfine field vector at the position of nucleus A and γ_A is the gyromagnetic ratio.

3.21.3 Simple example

The basic usage is very simple: One just needs a `%casscf` block and the simple input keyword `!DCD-CAS(2)` . The following example is a calculation on the LiF molecule. It possesses two singlet states that can be qualitatively described as ionic (Li^+ and F^-) and covalent (neutral Li with electron in 2s orbital and neutral F with hole in $2p_z$ orbital). At distances close to the equilibrium geometry, the ground state is ionic, while in the dissociation limit the ground state is neutral. Somewhere in between, there is an avoided crossing of the adiabatic potential energy curves where the character of the two states quickly changes (see figure Fig. 3.39 for potential energy curves for this system at the (QD)NEVPT2 level). At the CASSCF level, the neutral state is described better than the ionic state, with the result that the latter is too high in energy and the avoided crossing occurs at a too small interatomic distance. In the region where the avoided crossing actually takes place, the CASSCF states are then purely neutral or purely ionic. DCD-CAS(2) allows for a remixing of the states in the CASCI space under the effect of dynamic correlation, which will lower the ionic state more in energy than the neutral one. The input file is as follows:

```
! def2-TZVP DCD-CAS(2)
!moread

%moinp "casorbs.gbw" # guess with active orbitals in place

%casscf
  nel 2
  norb 2
  mult 1
  nroots 2
  actorbs locorbs
end

*xyz 0 1
Li 0.0 0.0 0.0
F 0.0 0.0 5.5
*
```

Since none of the standard guesses (`!PAtom`, `!PModel`, etc.) produces the correct active orbitals (Li 2s and F $2p_z$), we read them from the file casorbs.gbw. We also use the `actorbs locorbs` option to preserve the atomic character of the active orbitals and interpret the states in terms of neutral and ionic components easier. The following is the state composition of LiF at an interatomic distance of 5.5 angstrom at the CASSCF and DCD-CAS(2) levels.

```
---------------------------------------------
CAS-SCF STATES FOR BLOCK  1 MULT= 1 NROOTS= 2
---------------------------------------------

ROOT   0:  E=    -106.8043590118 Eh
```

(continues on next page)

(continued from previous page)

```
      0.99395 [     1]: 11
      0.00604 [     2]: 02
ROOT   1:  E=    -106.7485794535 Eh  1.518 eV  12242.2 cm**-1
      0.99396 [     2]: 02
      0.00604 [     1]: 11
```

```
---------------------------------------
      DCD-CAS(2) STATES
---------------------------------------

ROOT   0:  E=    -107.0917611937 Eh
      0.60590 [     2]: 02
      0.39410 [     1]: 11
ROOT   1:  E=    -107.0837717163 Eh  0.217 eV   1753.5 cm**-1
      0.60590 [     1]: 11
      0.39410 [     2]: 02
```

One can clearly see that while the CASSCF states are purely neutral (dominated by CFG `11`) or purely ionic (dominated by CFG `02`), the DCD-CAS(2) states are mixtures of neutral and ionic contributions. The calculation indicates that the interatomic distance of 5.5 is in the avoided crossing region. Note that the energies that are printed together with the DCD-CAS(2) state composition are the ones that are obtained from diagonalization of the DCD-CAS(2) dressed Hamiltonian. For excited states, these energies suffer from what we call *ground state bias* (see the original paper for a discussion [529]). A perturbative correction has been devised to overcome this problem. Our standard choice is first-order bias correction. The corrected energies are also printed in the output file and those energies should be used in production use of the DCD-CAS(2) method:

```
-------------------------------------------------------
 BIAS-CORRECTED (ORDER 1) STATE AND TRANSITION ENERGIES
=======================================================
 ROOT     Energy/a.u.    DE/a.u.     DE/eV     DE/cm**-1
=======================================================
   0:  -107.093214435   0.000000     0.000       0.0
   1:  -107.084988306   0.008226     0.224    1805.4
```

Warning

Note that the computational cost of a DCD-CAS(2) calculation scales as roughly the 3rd power of the size of the CASCI space. This makes calculations with active spaces containing more than a few hundred CSFs very expensive!

3.21.4 List of keywords

The following keywords can be used in conjunction with the DCD-CAS(2) method:

```
%casscf
  dcdcas true                 # Do a DCD-CAS(2) calculation
  diffded true                # Use difference-dedicated DCD-CAS(2) for the
                              # ij->ab class
  corrorder 1                 # Maximum order for the perturbative bias correction
                              # (lower orders come for free)
  dcd_ritrafo false           # Use RI approximation for electron repulsion␣
↪integrals

  dcd_soc false               # Relativistic DCD-CAS(2) with spin orbit coupling
  dcd_ssc false               # Relativistic DCD-CAS(2) with direct electronic
                              # spin-spin coupling
```

(continues on next page)

(continued from previous page)

```
  dcd_domagfield 0            # Number of user-specified finite magnetic fields
  dcd_dtensor false           # Calculate an effective Hamiltonian D-tensor
  dcd_nmultd 1                # The number of nonrelativistic multiplets for which␣
↪the
                              # D-tensor is calculated
  dcd_gmatrix false           # Calculate an effective Kramers pair Zeeman g-matrix
  dcd_amatrix false           # Calculate an effective Kramers pair Hyperfine A-
↪matrix
  dcd_kramerspairs 1          # The number of Kramers pairs for which g and/or A
                              # is calculated
  dcd_magnetization false     # Calculate the magnetization of the molecule in an
                              # external magnetic field

  dcd_cascimode false         # Run relativistic calculation in CASCI mode, i.e.
                              # without the dynamic correlation dressing
  dcd_natorbs false           # Calculate natural orbitals for each state and write
                              # them to disk
  dcd_populations false       # Perform population analysis on the DCD-CAS(2) states
end
```

Note that the calculation of SSC requires the definition of an auxiliary basis set, since it is only implemented in conjunction with RI integrals. If `dcd_magnetization` is requested, the values for magnetic flux density and temperature to be used can be specified via the keywords `MAGTemperatureMIN`, `MAGTemperatureMAX`, `MAGTemperatureNPoints`, `MAGFieldMIN`, `MAGFieldMAX`, `MAGNpoints` of the `rel` subblock of the `%casscf` block (see section *Magnetization and Magnetic Susceptibility*). If the keyword `dcd_domagfield` is set to a number different than 0, the magnetic fields can be entered as a matrix of xyz coordinates (in Gauss), e.g.

```
%casscf
  dcdcas true
  ...
  dcd_domagfield 2
  dcd_magneticfields[0] = 10000, 0, 0
  dcd_magneticfields[1] = 0, 10000, 0
end
```

Furthermore, there is the keyword `DCD_EDIAG` that when running the DCD-CAS(2) code in CASCI mode works analogously to the keyword `EDiag` in the `soc` subblock of the `%mrci` block (see section *Zero-Field Splitting*). The energies should be entered in atomic units, e.g.

```
%casscf
  ...
  dcdcas true
  dcd_cascimode true
  dcd_soc true
  DCD_EDIAG[0] -2220.920028
  DCD_EDIAG[1] -2220.834377
  DCD_EDIAG[2] -2220.835871
  DCD_EDIAG[3] -2220.810290
  DCD_EDIAG[4] -2220.812293
  DCD_EDIAG[5] -2220.756732
end
```

3.22 Full Configuration Interaction

ORCA provides several exact and approximate approaches to tackle the full configuration interaction (FCI) problem. These methods are accessible via the CASSCF module (see Section *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)*) or the ICE module (described in Section *Approximate Full CI Calculations in Subspace: ICE-CI*).

In the following, we compute the FCI energy of the lithium hydride molecule using the CASSCF module, where a typical input requires the declaration of an active space. The latter defines the number of active electron and orbitals, which are evaluated with the FCI ansatz. In the special case that all electrons and orbitals are treated with the FCI ansatz, we can use the keyword `DoFCI` in the `%CASSCF` block and let the program set the active space accordingly. In this example, we focus on the singlet ground state. Note that excited states for arbitrary multiplicities can be computed with the keywords `Mult` and `NRoots`. The FCI approach is invariant to orbital rotations and thus orbital optimization is skipped in the CASSCF module. Nevertheless, it is important to employ a set of meaningful orbitals, e.g. from a converged Hartree-Fock calculation, to reduce the number of FCI iterations.

```
# Hartree-Fock orbitals
!def2-tzvp RHF

*xyz 0 1
  Li 0 0 0
  H  0 0 1.597
*
```

The output of the Hartree-Fock calculation also reports on the total number of electrons and orbitals in your system (see snippet below).

```
Number of Electrons    NEL              ....    4
Basis Dimension        Dim              ....   20
```

In the given example, there are 4 electrons in 20 orbitals, which is a "CAS(4,20)". Reading the converged RHF orbitals, we can start the FCI calculation.

```
!def2-tzvp extremescf

!moread
%moinp "RHF.gbw"

%maxcore 2000

%casscf
  DoFCI true # sets NEL 4 and NORB 20 in this example.
end

*xyz  0 1
  Li 0 0 0
  H  0 0 1.597
*
```

The output reports on the detailed CI settings, the number of configuration state functions (CSFs) and the CI convergence thresholds.

```
CI-STEP:
  CI strategy                          ... General CI
  Number of multiplicity blocks        ...    1
  BLOCK  1 WEIGHT=   1.0000
  Multiplicity                       ...    1
  #(Configurations)                  ... 8455
  #(CSFs)                            ... 13300
  #(Roots)                           ...    1
  ROOT=0 WEIGHT=    1.000000
```

(continues on next page)

(continued from previous page)

```
  PrintLevel                        ...    1
  N(GuessMat)                       ...       512
  MaxDim(CI)                        ...        10
  MaxIter(CI)                       ...        64
  Energy Tolerance CI               ...  1.00e-13
  Residual Tolerance CI             ...  1.00e-13
  Shift(CI)                         ...  1.00e-04
  ...
```

The program then prints the actual CI iterations, the final energy, and the composition of the wave function in terms of configurations (CFGs).

```
------------------
CAS-SCF ITERATIONS
------------------


  MACRO-ITERATION   1:
  --- Inactive Energy E0 = 0.99407115 Eh
  --- All densities will be recomputed
  CI-ITERATION   0:
  -8.012799617   0.526896429727 (    0.25)
  CI-ITERATION   1:
  -8.047996328   0.001601312242 (    0.25)
  CI-ITERATION   2:
  -8.048134967   0.000022625293 (    0.25)
  CI-ITERATION   3:
  -8.048137773   0.000000462227 (    0.25)
  CI-ITERATION   4:
  -8.048137841   0.000000035496 (    0.25)
  CI-ITERATION   5:
  -8.048137845   0.000000001357 (    0.25)
  CI-ITERATION   6:
  -8.048137845   0.000000000254 (    0.25)
  CI-ITERATION   7:
  -8.048137845   0.000000000006 (    0.25)
  CI-ITERATION   8:
  -8.048137845   0.000000000001 (    0.25)
  CI-ITERATION   9:
  -8.048137845   0.000000000000 (    0.25)
  CI-PROBLEM SOLVED
  DENSITIES MADE

  <<<<<<<<<<<<<<<<<<INITIAL CI STATE CHECK>>>>>>>>>>>>>>>>>>

  BLOCK  1 MULT= 1 NROOTS= 1
  ROOT   0:  E=      -8.0481378449 Eh
  0.97242 [     0]: 22000000000000000000
  0.00296 [    99]: 20000002000000000000
  0.00258 [    89]: 20000010001000000000
  0.00252 [    85]: 20000020000000000000
```

Aside from energies, the CASSCF module offers a number of properties (g-tensors, ZFS, …), which are described in Section *CASSCF Properties*.

The exact solution of the FCI problem has very steep scaling and is thus limited to smaller problems (at most active spaces of 16 electrons in 16 orbitals). Larger systems are accessible with approximate solutions, e.g. with the density matrix renormalization group approach (DMRG), described in Section *Density Matrix Renormalization Group (DMRG)*, or the iterative configuration expansion (ICE) reported in Section *Approximate Full CI Calculations in Subspace: ICE-CI*. For fun, we repeat the calculation with the ICE-CI ansatz, which offers a more traditional approach to get an approximate full CI result.

```
!def2-TZVP ExtremeSCF

!moread
%moinp "RHF.gbw"

%maxcore 2000

%ice
  Nel 4
  Norb 20
end

*xyz  0 1
  Li 0 0 0
  H  0 0 1.597
*
```

The single most important parameter to control the accuracy is `TGen`. It is printed with the more refined settings in the output. We note passing that the wave function expansion and its truncation can be carried out in the basis of CSFs, configurations, or determinants. The different strategies are discussed in detail by Chilkuri *et al.* [443, 444].

```
ICE-CI:
  General Strategy                ... CONFIGURATIONS (all CSFs to a given CFG,␣
↪spin adapted)
  Max. no of macroiterations      ...   12
  Variational selection threshold ... -1.000e-07
  negative! => TVar will be set to 1.000e-07*Tgen=1.000e-11
  Generator selection threshold   ... 1.000e-04
  Excitation level                ... 2
  Selection on initial CSF list   ... YES
  Selection on later CSFs lists   ... YES

  ...

******************************
*  ICECI MACROITERATION   3  *
******************************

# of active configurations =   2808
Initializing the CI                ...  (CI/Run=3,2 UseCC=0)done (  0.0 sec)
Building coupling coefficients     ... (CI/Run=3,2)Calling BuildCouplings_RI␣
↪UseCCLib=0 DoRISX=0
CI_BuildCouplings NCFG= 2808 NORB=20 NEL=4 UseCCLib=0 MaxCore=2000
PASS 1 completed. NCFG= 2808 NCFGK= 8416 MaxNSOMOI=4 MaxNSOMOK=4
PASS 2 completed.
PASS 3 completed.
Memory used for RI tree            =    2.99 MB (av. dim=    35)
Memory used for ONE tree           =    1.32 MB (av. dim=    46)
Memory used for coupling coefficients=    0.01 MB
done (    0 sec)
Now calling CI solver (4095 CSFs)

****Iteration    0****
Maximum residual norm  :     0.000293130557

****Iteration    1****
Maximum residual norm  :     0.000000565920

****Iteration    2****
Maximum residual norm  :     0.000001755176

****Iteration    3****
```

(continues on next page)

(continued from previous page)

```
Maximum residual norm  :     0.000000435942
Rebuilding the expansion space

****Iteration    4****

*** CONVERGENCE OF ENERGIES REACHED ***
CI problem solved in   0.4 sec

CI SOLUTION :
STATE   0 MULT= 1: E=     -8.0481340246 Eh W=  1.0000 DE= 0.000 eV      0.0 cm**-1
0.97249  : 22000000000000000000
Selecting new configurations         ... (CI/Run=3,2)done (  0.0 sec)
# of selected configurations          ...   2747
# of generator configurations         ...     43 (NEW=1 (CREF=43))
Performing single and double excitations relative to generators ... done (  0.0␣
↪sec)
# of configurations after S+D         ...   7038
Selecting from the generated configurations  ... done (  0.1 sec)
# of configurations after Selection  ...   2808
Root   0:     -8.048134025       -0.000000023      -8.048134048
==>>> CI space seems to have converged. No new configurations
maximum energy change                ... 1.727e-05 Eh

               ********* ICECI IS CONVERGED *********
  Initializing the CI                  ...  (CI/Run=3,3 UseCC=0)done (  0.0 sec)
  Building coupling coefficients       ... (CI/Run=3,3)Calling BuildCouplings_RI␣
↪UseCCLib=0 DoRISX=
  CI_BuildCouplings NCFG= 2808 NORB=20 NEL=4 UseCCLib=0 MaxCore=2000
  PASS 1 completed. NCFG= 2808 NCFGK= 8416 MaxNSOMOI=4 MaxNSOMOK=4
  PASS 2 completed.
  PASS 3 completed.
  Memory used for RI tree              =    2.99 MB (av. dim=    35)
  Memory used for ONE tree             =    1.32 MB (av. dim=    46)
  Memory used for coupling coefficients=    0.01 MB
  done (    0 sec)
  Now calling CI solver (4095 CSFs)

  ****Iteration    0****
  Maximum residual norm  :     0.000000471011

  ****Iteration    1****

  *** CONVERGENCE OF ENERGIES REACHED ***
  CI problem solved in   0.1 sec

  CI SOLUTION :
  STATE   0 MULT= 1: E=     -8.0481340245 Eh W=  1.0000 DE= 0.000 eV      0.0 cm**-
↪1
  0.97249  : 22000000000000000000
```

With Hartree-Fock orbitals and the default settings, the ICE converges in 3 macro iterations to an energy of $-8.048134047513\ E_\mathrm{h}$. The deviation from the exact solution is just $3.8 \times 10^{-6}\ E_\mathrm{h}$ in this example.

3.23 Molecular Mechanics

Since version 4.2 ORCA features its own independent Molecular Mechanics (MM) implementation.

The minimum input necessary for an MM treatment looks as follows.

```
!MM
%mm
 ORCAFFFilename "UBQ.ORCAFF.prms"
end
```

In this section we discuss usage of MM as well as basic keywords and options, i.e.

- with which other features MM calculations can be combined
- the basic structure of the ORCA Force Field File,
- which force fields are supported,
- how to generate the ORCA Force Field File,
- how to manipulate the ORCA Force Field File,
- how to speed up MM calculations,
- further MM options and keywords.

Further options important for QM/MM calculations will be discussed in section *Overview on ORCA's Multiscale Implementation*.

3.23.1 Combining the MM module with other ORCA Features

Molecular mechanics was introduced as a prerequisite for multiscale simulations within ORCA. But MM calculations can also be used as a standalone, e.g. for cheap preoptomizations.

Possible ORCA features are:

- **Single Point Calculations**: Simple MM calculations as well as *QM/MM calculations*
- **Optimization**: Use all kinds of *geometry optimizations* with all kinds of constraints, TS optimization, relaxed surface scans, and the ScanTS feature. Use the L-Opt and L-OptH features including the combination of all kinds of fragment optimizations (fix fragments, relax fragments, relax only specific elements in fragments, treat a fragment as a rigid body).
- **Transition States and Minimum Energy Paths**: Bond breaking and bond forming are not possible using MM methods, but conformational changes or dispersion driven reactions can be studied on MM-level only. For such tasks ORCA *Nudged-Elastic Band method* can be used.
- **Conformational Searches**: Conformational Searches can be run with ORCA's *GOAT* module.
- **Frequency Calculations**: Use regular *frequency calculations*. If required, ORCA automatically switches on the *Partial Hessian Vibrational Analysis* (PHVA) [531] calculation. (not tested for `IONIC-CRYSTAL-QMMM`)
- **Molecular Dynamics**: Use the *Molecular Dynamics* (MD) module for Born-Oppenheimer MD (BOMD) with MM.

> **Important**
>
> *Active regions* can also be defined for MM calculations. They have to be *defined* via the `%qmmm` block.

3.23.2 Molecular Mechanics Force Field in ORCA

For the MM part of the QM/MM calculation force-field parameters are necessary. ORCA has its own parameter file format (ORCA force field file - ORCAFF.prms), which includes the atom specific parameters for nonbonded interactions:

- partial charge
- LJ coefficients

and parameters for bonded interactions:

- bonds
- angles
- Urey-Bradley terms
- dihedrals
- impropers
- CMAP terms (cross-terms for backbone, currently not used)

Individual parameters, like e.g. atomic charge, equilibrium bond length and force constant, …, can be conveniently modified directly within the ORCA Force Field File.

Supported Force Field Types

For the MM part of the QM/MM calculation force field parameters are necessary. ORCA has its own force field file format (stored in files called `basename.ORCAFF.prms`), which is used internally. ORCA provides tools to convert other popular force fields into the ORCA force field file format. Currently, the following formats are supported to be converted to the ORCA force field file format:

- **CHARMM psf files**: protein structure file from the CHARMM force field. The psf files can be easily prepared with the popular molecular visualizer VMD, together with its extensions (psfgen, QwikMD, fftk, which works together with ORCA). The psf file contains information on the atom types and on the bonded interactions of all atoms. It does, however, not contain the parameters that belong to these interactions. These parameteres are stored in specific files, often ending with prm, but also par or str. The CHARMM parameter files come with VMD installation, can be directly downloaded.
- **AMBER prmtop files**: topology files from the AMBER force field. Tutorials on how to generate AMBER prmtop files (for standard and non-standard molecules) can be found here.
- **Open Force Field**: xml files from the openforcefield initiative. With the openff-toolkit xml topology files (compatible with the AMBER force field) can be easily generated for small and medium-sized non-standard molecules. For a tutorial see here.
- **Simple force field** for small to medium-sized molecules: Alternatively, the orca_mm module can generate a simple approximate ORCAFF.prms file. For more options, see *Create simple force field: makeff*.

This concept has the following advantages:

- Modification of force field parameters: Atom and bond specific parameters can be easily modified within the ORCA force field file, allowing the user maximum flexibility in modifying the force field, which might be particularly useful for chemical systems like transition metal complexes.
- Standard and Non-Standard Ligands: Ligands can be easily and flexibly exchanged or added to the system, see *How to generate and manipulate the ORCA Force Field File*.

How to generate and manipulate the ORCA Force Field File

Once a ORCAFF.prms file is available, it can be manipulated, i.e. split up into several parts for individual molecules, new ORCAFF.prms files can be generated for non-standard molecules, and individual ORCAFF.prms files can be merged, as described in the following:

Conversion from other formats to ORCAFF.prms: convff

The orca_mm module can convert psf and prm files (CHARMM), prmtop files (AMBER) or xml files (open force field from the openff toolkit, compatible to AMBER) to the ORCAFF file with the -convff flag. Input options are:

```
orca_mm -convff <optional:-verbose> <FFInput> <PSFFILE> <PRMFILE(S)>

Keywords:
          <FFInput> = -CHARMM
          <FFInput> = -AMBER
          <FFInput> = -OPENMM
```

For CHARMM topologies, when a psf file is available for a system with standard residues, prepared by e.g. QwikMD, psfgen or other vmd tools, the conversion needs the psf plus the prm files as input:

```
CHARMM example:
orca_mm -convff -CHARMM 1C1E.psf par_all36_prot.prm toppar_water_ions_namd.str
```

ORCA can also convert Amber topologies to the ORCAFF file. Here, only the prmtop file is required:

```
AMBER example:
orca_mm -convff -AMBER complex.prmtop
```

ORCA can also convert xml files from the openff toolkit (AMBER compatible) to the ORCAFF file. Here, only the xml file is required:

```
OPENFF example:
orca_mm -convff -OPENMM complex.xml
```

Divide a force field file: splitff

If a ORCAFF.prms file should be subdivided into several files, e.g. if the psf file stems from QWikMD with non-standard molecules included, e.g. a ligand. In that case first the parameters of the ligand are split from the remaining system, next the ligand needs to be protonated, then a simple ORCAFF.prms file is generated via orca_mm's makeff option, and finally the ligand's new ORCAFF.prms file is added to the main systems file via the above described mergeff option. Note that the file can only be split into files for nonbonded fragments.

Input options:

```
orca_mm -splitff <optional:-verbose> <ORCAFFFILE> <A1> <optional:A2> ...

Keywords:

          <ORCAFFFILE>  = ORCA forcefield file.
          <A1>          = Atom number of first atom of fragment that should belong
                             to a new ORCA forcefield file
          <A2>          = Atom number of first atom of fragment that should belong
                             to a new ORCA forcefield file
          ...           = More split atoms possible
          Note that atoms start counting at 1.

Example:
orca_mm -splitff 1C1E_substrate_noH.ORCAFF.prms 7208
```

Merge force field files: mergeff

If several ORCAFF.prms files are available and should be merged for an ORCA calculation, e.g. for a standard plus a non-standard molecule.

Input options:

```
orca_mm -mergeff <optional:-verbose> <ORCAFFFILE1> <ORCAFFFILE2> ...

Keywords:

          <ORCAFFFILE1>   = First ORCA forcefield file
          <ORCAFFFILE2>   = Second ORCA forcefield file
          ...             = More ORCA forcefield files possible

Example:
orca_mm -mergeff 1C1E.ORCAFF.prms substrate_withH.ORCAFF.prms
```

Repeat force field files: repeatff

In case the same ORCAFF.prms file needs to be repeated multiple times, the repeatff functionality is available.

Input options:

```
orca_mm -repeatff <optional:-verbose> <ORCAFFFILE> <A>
          <ORCAFFFILE>      = ORCA forcefield file.
          <A>               = Factor (integer) defining how often this forcefield␣
↪file should be repeated.

Example:
orca_mm -mergeff methanol.ORCAFF.prms 580
```

This feature can be useful e.g. in the case of solvating a molecule, i.e. adding multiple copies of a solvent to a solute. First the solvent can be repeated N times, and subsequently the solute's prms file can be merged together with the solvent prms file.

Divide a PDB file: splitpdb

When splitting a ORCAFF.prms file, also splitting of the pdb file is required. The file can be split into an arbitrary number of individual files.

This can be useful together with the splitff command.

Input options:

```
orca_mm -splitpdb <optional:-verbose> <PDBFILE> <A1> <optional:A2> ...

Keywords:

          <PDBFILE> = PDB file.
          <A1>      = Atom number of first atom of fragment that should belong
                      to a new ORCA forcefield file
          <A2>      = Atom number of first atom of fragment that should belong
                      to a new ORCA forcefield file
          ...       = More split atoms possible
          Note that atoms start counting at 1.

Example:
orca_mm -splitpdb 1C1E_substrate_noH.pdb 7208
```

Merge PDB files: mergepdb

If several PDB files are available and need to be merged for an ORCA calculation, e.g. a protein and a ligand or multiple ligands, or a ligand that was first removed from a complex, then modified, and finally should get back into the complex PDB file.

This can be useful together with the mergeff command.

Input options:

```
orca_mm -mergepdb <optional:-verbose> <1PDBFILE> <2PDBFILE> ...

Keywords:

          <1PDBFILE>   = First PDB file
          <2PDBFILE>   = Second PDB file
          ...          = More PDB files possible

Example:
orca_mm -mergepdb 1C1E.pdb substrate_withH.pdb
```

Create simple force field: makeff

The orca_mm tool can generate an approximate force field for a molecule, storing it in ORCAFF.prms format. Here, the LJ coefficients are based on UFF parameters and the partial charges are based on a simple PBE or XTB calculation. The resulting topology is certainly not as accurate as an original CHARMM topology, but can still be used for an approximate handling of the molecule. Herewith, the molecule can be part of the QM region (having at least the necessary LJ coefficients), or part of the MM region as a non-active spectator - being not too close to the region of interest. In the latter case it is important that the molecule is not active, since bonded parameters are not available. However, it can still be optimized as a rigid body, see sections *Geometry Optimizations using the L-BFGS optimizer* and the usage in QM/MM calculations in section *Optimization with the Cartesian L-BFGS Minimizer*, on MM level, optimizing its position and orientation with respect to the specific environment.

Input options:

```
orca_mm -makeff <optional:-verbose> <XYZ/PDBFILE> <optional:-C CHARGE>
                <optional:-M MULT> <optional:-nproc N> <optional:-CHARGE_OPTIONS>

Keywords:

          <CHARGE>              = charge of system
          <MULT>                = multiplicity of system
          <-nproc N>            = number of processors (Default 1)
          <-CHARGE_OPTIONS>     =   Structure            Charge calculation
               <-PBE>                input                     PBE
               <-PBEOpt>             PBE opt.                  PBE
               <-PBEOptH>            PBE H-opt.                PBE
               <-XTB>                input                     XTB
               <-XTBOpt>             XTB opt.                  XTB
               <-XTBOptH>            XTB H-opt.                XTB
               <-XTBOptPBE>          XTB opt.                  PBE
               <-noChargeCalc>       distribute net charge evenly

               PBE Opt and SP level: RI-PBE/def2-SVP CPCM(Water), CHELPG charges
               XTB Opt and SP level: GFN2-XTB GBSA(Water), Mulliken partial charges

Example:
orca_mm -makeff substrate_withH.xyz -M 2 -XTBOptPBE
```

Note that ORCA generates bonds based on simple distance rules, which enables ORCA to detect where to add link atoms between QM and MM atoms, see also section *Boundary Region*. But the user is advised to treat a molecule,

for which the ORCAFF.prms file was generated with the makeff option, either fully in the QM, or fully in the MM region, unless the charge distribution has been properly taken care of (due to the need of integer charges in QM and MM system).

Get standard hydrogen bond lengths: getHDist

For the definition of the link atoms standard bond lengths between C, N and O and hydrogen are directly set by ORCA but can be modified by the user, see section *Boundary Region*. If other atom types are on the QM side of the QM-MM boundary, their distance to the link atom has to be defined. In this case a file can be provided to ORCA which defines the standard bond length to hydrogen for all possible atoms. Such a file can be generated via the following command:

Input options:

```
orca_mm -getHDist <optional:-verbose> <XYZ/PDBFILE>

Example:
orca_mm -getHDist 1C1E.xyz
```

This file can then be modified, the required values can be added, and the resulting file can be defined as input for the QMMM calculation.

Create ORCAFF.prms file for IONIC-CRYSTAL-QMMM

For IONIC-CRYSTAL-QMMM calculations, section *IONIC-CRYSTAL-QMMM*, an ORCAFF.prms file with initial charges and connectivities is required. If you are not using the `orca_crystalprep` tool for setting up such calculations, see section *orca_crystalprep*, you can directly prepare the ORCAFF.prms file with the command:

```
orca_mm -makeff <XYZ/PDBFILE> -CEL <ELEMENT1> <OXIDATION_STATE1>
                              -CEL <ELEMENT2> <OXIDATION_STATE2>

Keywords:

          <ELEMENT1>              = element name
          <OXIDATION_STATE1>      = formal oxidation state of element 1
          ...                     = More elements possible

Example:
orca_mm -makeff na4cl4.xyz -CEL Na 1.0 -CEL Cl -1.0
```

Here, na4cl4.xyz is the supercell structure file (it can contain tens of thousands of atoms).

> **Note**
>
> For supercells with more complex oxidation states', e.g. Co_3O_4, the ORCAFF.prms file can be generated conveniently via the *orca_crystalprep* tool.

3.23.3 Speeding Up Nonbonded Interaction Calculation

For MM calculations of very large systems with hundreds of thousands of atoms, and for QMMM calculations with fast QM methods (e.g. XTB, AM1) and / or very small QM systems, the computation of the nonbonded interactions can become a bottleneck. Different schemes for speeding up the calculation of nonbonded interactions are available within the ORCA MM implementation. Two schemes truncate long-range interaction, another scheme can be used for calculations with active regions, i.e. calculations where only a part of the system is active or optimized. For more information on active regions see section *Active Atoms in Combination with Optimization, Frequency Calculation, MD*.

Force Switching for LJ Interaction

With force switching for the LJ interaction (described in reference [532]) a smooth switching function is used to truncate the LJ potential energy smoothly at the outer cutoff distance `LJCutOffOuter`. If switching is set to false, the LJ interaction is not truncated at `LJCutOffOuter`. The parameter `LJCutOffInner` specifies the distance at which the switching function starts taking an effect to bring the van der Waals potential to 0 smoothly at the `LJCutOffOuter` distance, ensuring that the force goes down to zero at `LJCutOffOuter`, without introducing discontinuities. Note that `LJCutOffInner` must always be smaller than `LJCutOffOuter`.

```
%mm
 SwitchForceLJ true       # Use the switch force scheme for the LJ interaction.
                          # Default: true.
 LJCutOffInner 10.        # Distance (in Ang). Default: 10 Ang.
 LJCutOffOuter 12.        # Distance (in Ang). Default: 12 Ang.
end
```

Force Shifting for Electrostatic Interaction

With force shifting for the electrostatic interaction (described in reference [532]) the electrostatic potential is shifted to zero at the cutoff distance `CoulombCutOff`. If shifting is set to false, the electrostatic interaction is not truncated at `CoulombCutOff`.

```
%mm
 ShiftForceCoulomb true   # Use the shift force scheme for the Coulomb interaction.
                          # Default: true.
 CoulombCutOff 12.        # Distance (in Ang). Default: 12 Ang.
end
```

Neglecting Nonbonded Interactions Within Non-Active Region

When using active regions (see section *Active Atoms in Combination with Optimization, Frequency Calculation, MD*) for optimizations and MD runs, the nonbonded interactions at the MM level can be neglected for those atom pairs, which are both non-active, without loss of accuracy for the results. Even relative energies between two structures are correct, if the atom positions of the non-active atoms are identical. For all other cases, i.e. if the positions of atoms in the non-active region differ, the full nonbonded interaction should be computed in the final single-point energy calculation. By default this option is switched off.

```
%mm
 Do_NB_For_Fixed_Fixed true # Compute MM-MM nonbonded interaction also for
                            # non-active atom pairs. Default true.
end
```

3.23.4 Rigid Waters

As TIP3P water might have to be treated as rigid bodies due to its parametrization - please check out the specifics of the applied force field parametrization - we offer a keyword for the automated rigid treatment of all active MM waters. The following keyword applies bond and angle constraints to active MM waters in optimization as well as MD runs:

```
%mm
 Rigid_MM_Water false       # Default: false.
end
```

3.23.5 Available Keywords for the MM module

Here we list all keywords that are accessible from within the mm block and that are relevant to MM calculations.

Table 3.57: `%mm` block input keywords.

Keyword	Options	Description
`ShiftF`	`true`	Long range electrostatics: Use the Shift Force scheme. Default true. The Shift Force scheme for the Coulomb interaction shifts the Coulomb potential such that it becomes zero at the cutoff distance.
`Coulom`	`<rea`	Distance (in Ang). Default: 12 Ang.
`Switch`	`true`	Use the Switch Force scheme. Default true. With the Switch Force scheme for the LJ interaction is unchanged up to `LJCutOffInner`. Between `LJCutOffInner` and `LJCutOffOuter` a smooth switching function is applied onto the LJ potential so that the force goes down to zero at `LJCutOffOuter`, without introducing discontinuities.
`LJCutO`	`<rea`	Distance (in Ang). Default: 10 Ang.
`LJCutO`	`<rea`	Distance (in Ang). Default: 12 Ang.
`Dielec`	`<rea`	Dielectric constant used in calc. of electrostatic interaction. Default: 1.
`Coulom`	`<rea`	Scaling factor for electrostatic interaction between 1,4-bonded atoms. Default: 1.
`PrintI`	`<int`	Printing options: Can be 0 to 4, 0=nothing, 1=normal, …

Some of the MM keywords can also be accessed via the qmmm block, see section *Additional Keywords*.

Table 3.58: `%mm` block input keywords which can also be accessed from the `%qmmm` block.

Keyword	Options	Description
`ORCAFFFilen`	`"<filen`	Filename of the ORCA Force Field prms file.
`Do_NB_For_F`	`true/fa`	Compute MM-MM nonbonded interaction also within non-active region, i.e. for non-active atom pairs. Default `true`. For `GOAT` and `Crystal-QMMM` default is `false`.
`RIGID_MM_WA`	`true/fa`	Optimization and MD of active MM waters. Default `false`.
`ExtendActiv`	`distanc`	Scheme to extend the active region to activeRegionExt. See also *here*. `distance`: Use a distance criterion (VDW distance plus `Dist_AtomsAroundOpt`).
	`cov_bon`	Add only atoms bonded covalently to active atoms.
	`no`	Do not add any atoms to activeRegionExt.
`Dist_AtomsA`	`<real>`	Distance to be added to VDW distance for `ExtendActiveRegion`. In Angstrom. Default `0`.
`OptRegion_F`	`{ints}`	List of atoms to define the extended active region. Default: empty list.
`PrintOptReg`	`true/fa`	Additionally print xyz and trj for opt region. Default `true`.
`PrintOptReg`	`true/fa`	Additionally print xyz and trj for extended opt region. Default `false`.
`PrintPDB`	`true/fa`	Additionally print pdb file for entire system, is updated every iteration for optimization. Default `false`.

CHAPTER

FOUR

STRUCTURE AND REACTIVITY

4.1 Geometry Optimizations

ORCA is able to calculate equilibrium structures (minima and transition states) using the quasi Newton update procedure with the well known BFGS update [533, 534, 535, 536, 537, 538], the Powell or the Bofill update. The optimization can be carried out in either redundant internal (recommended in most cases) or Cartesian displacement coordinates. As initial Hessian the user can choose between a diagonal initial Hessian, several model Hessians (Swart, Lindh, Almloef, Schlegel), an exact Hessian and a partially exact Hessian (both recommended for transition state optimization) for both coordinate types. In redundant internal coordinates several options for the type of step to be taken exist. The user can define constraints via two different paths. He can either define them directly (as bond length, angle, dihedral or Cartesian constraints) or he can define several fragments and constrain the fragments internally and with respect to other fragments. The ORCA optimizer can be used as an external optimizer, i.e.the energy and gradient calculations done by ORCA.

The usage of analytic gradients is necessary for efficient geometry optimization. In ORCA 5.0, the following methods provide analytic first derivatives

- Hartree-Fock (HF) and DFT (including the RI, RIJK and RIJCOSX approximations)
- MP2, RI-MP2 and DLPNO-MP2
- TD-DFT for excited states
- CAS-SCF

When the analytic gradients are not available, it is possible to evaluate the first derivatives numerically by finite displacements. This is available for all methods.

Some methods for locating transition states (TS) require second derivative matrices (Hessian), implemented analytically for HF, DFT and MP2. Additionally, several approaches to construct an initial approximate Hessian for TS optimization are available. A very useful feature for locating complicated TSs is the Nudged-Elastic Band method in combination with the TS finding algorithm (NEB-TS, ZOOM-NEB-TS). An essential feature for chemical processes involving excited states is the conical intersection optimizer. Another interesting feature are MECP (Minimum Energy Crossing Point) optimizations.

For very large systems ORCA provides a very efficient L-BFGS optimizer, which makes use of the orca_md module. It can also be invoked via simple keywords described at the end of this section.

4.1.1 Geometry Optimizations

The use of the geometry optimization module is relatively straightforward.[1] Here are some typical examples:

```
! BLYP SVP Opt
* int 0 1
    C  0 0 0 0.0000   0.000    0.00
    O  1 0 0 1.2029   0.000    0.00
    H  1 2 0 1.1075 122.016    0.00
    H  1 2 3 1.1075 122.016  180.00
*
```

ORCA employs the *RIJ* by default. If this is not wanted, RI can be deactivated generally by using the `!NORI` keyword.

An optimization of the first excited state of ethylene:

```
! BLYP SVP OPT

%tddft
   IRoot 1
end

* xyz 0 1
 C     0.000000     0.000000     0.666723
 C     0.000000     0.000000    -0.666723
 H     0.000000    -0.928802     1.141480
 H    -0.804366    -0.464401    -1.341480
 H     0.000000     0.928802     1.241480
 H     0.804366     0.464401    -1.241480
*
```

4.1.2 Initial Hessian for Minimization

The convergence of a geometry optimization crucially depends on the quality of the initial Hessian. In the simplest case it is taken as a unit matrix (in redundant internal coordinates we use 0.5 for bonds, 0.2 for angles and 0.1 for dihedrals and improper torsions). However, simple model force-fields like the ones proposed by Schlegel, Lindh, Swart or Almlöf are available and lead to much better convergence. The different guess Hessians can be set via the `InHess` option which can be either `unit`, `Almloef`, `Lindh`, `Swart` or `Schlegel` in redundant internal coordinates, furhter shown in the keywords list found within Section *Geometry Optimization Keywords*. Since version 2.5.30, these model force-fields (built up in internal coordinates) can also be used in optimizations in Cartesian coordinates.

For minimizations we recommend the `Almloef` Hessian, which is the default for minimizations. The `Lindh` and `Schlegel` Hessian yield a similar convergence behaviour. There is also the option for the `Swart` model Hessian, which is less parametrized and should improve for weakly interacting and/or unusual structures. Of course the best Hessian is the exact one. `Read` may be used to input an exact Hessian or one that has been calculated at a lower level of theory (or a "faster" level of theory). From version 2.5.30 on this option is also available in redundant internal coordinates. But we have to point out that the use of the exact Hessian as initial one is only of little help, since in these cases the convergence is usually only slightly faster, while at the same time much more time is spent in the calculation of the initial Hessian.

To sum it up: we advise to use one of the simple model force-fields for minimizations.

[1] But that doesn't mean that geometry optimization itself is straightforward! Sometimes, even when it is not expected the convergence can be pretty bad and it may take a better starting structure to come to a stationary point. In particular floppy structures with many possible rotations around single bonds and soft dihedral angle modes are tricky. It may sometimes be advantageous to compute a Hessian matrix at a "cheap" level of theory and then do the optimization in Cartesian coordinates starting from the calculated Hessian.

4.1.3 Coordinate Systems for Optimizations

The coordinate system for the optimization can be chosen by the `coordsys` variable that can be set to `cartesian` or `redundant`within the `%geom` block. The default is the redundant internal coordinate system. If the optimization with `redundant` fails, you can still try `cartesian`. If the optimization is then carried out in Cartesian displacement coordinates with a simple model force-field Hessian, the convergence will be only slightly slower. With a unit matrix initial Hessian very slow convergence will result.

A compound job `two_step_opt.inp` that first computes a semi-empirical Hessian to start from is shown below:

```
* int 0 1
C 0 0 0 0 0 0
O 1 0 0 1.3 0 0
H 1 2 0 1.1 110 0
H 1 2 3 1.1 110 180
*

%compound
# Step 1: semiempirical calculation of the Hessian
New_Step
! AM1 NumFreq
Step_End

# Step 2: optimization starting from previous Hessian
New_Step
!B3LYP def2-svp def2/J Opt
%geom
InHess
 Read
InHessName "two_step_opt_Compound_1.hess"
# this file must be either a .hess file from a
# frequency run or, a .opt/.carthess file left over from a
# previous geometry optimization
end
Step_End

End
```

Tip

- For transition metal complexes MNDO, AM1 or PM3 Hessians are not available. You can use ZINDO/1 or NDDO/1 Hessians instead. They are of lower quality than MNDO, AM1 or PM3 for organic molecules but they are still far better than the standard unit matrix choice.
- If the quality of the initial semi-empirical Hessian is not sufficient you may use a "quick" RI-DFT job (e.g. `BP def2-sv(p) defgrid1`)
- In semi-empirical geometry optimizations on larger molecules or in general when the molecules become larger the redundant internal space may become large and the relaxation step may take a significant fraction of the total computing time.

For condensed molecular systems and folded molecules (e.g. a U-shaped carbon chain) atoms can get very close in space, while they are distant in terms of number of bonds connecting them. As damping of optimization steps in internal coordinates might not work well for these cases, convergence can slow down. ORCA's automatic internal coordinate generation takes care of this problem by assigning bonds to atom pairs that are close in real space, but distant in terms of number of bonds connecting them. See keywords `AddExtraBonds`, `AddExtraBonds_MaxLength`, and `AddExtraBonds_MaxDist` within the keyword list found in Section *Geometry Optimization Keywords*.

For solid systems modeled as embedded solids the automatically generated set of internal coordinates might become very large, rendering the computing time spent in the optimization routine unnecessarily large. Usually, in such calculations the cartesian positions of outer atoms, coreless ECPs and point charges are constrained during the opti-

mization - thus most of their internal coordinates are not needed. By requesting `ReduceRedInts true` within the `%geom` block only the required needed internal coordinates (of the constrained atoms) are generated.

OBS: If the step in `redundant` fails badly and only Cartesian constrains are set (or no constrains), ORCA will fallback to a `cartesian` step automatically. This can be turned off by setting CARTFALLBACK to FALSE.

Redundant Internal Coordinates

There are three types of internal coordinates: redundant internals, old redundant internals (redundant_old) and a new set of redundant internals (redundant_new, with improved internals for nonbonded systems). All three sets work with the same "primitive" space of internal coordinates (stretches, bends, dihedral angles and improper torsions). Only the redundant internals works with one more type of bends in cases where a normal bend would have been approximately $180°$. In redundant internal coordinates the full primitive set is kept and the Hessian and gradient are transformed into this – potentially large – space. A geometry optimization step requires, depending on the method used for the geometry update, perhaps a diagonalization or inversion of the Hessian of dimension equal to the number of variables in the optimization. In redundant internal coordinates this space may be 2-4 times larger than the nonredundant subspace which is of dimension $3N_{\text{atoms}} - 6(5)$. Since the diagonalization or inversion scales cubically the computational overhead over nonredundant spaces may easily reach a factor of 8–64. Thus, in redundant internal coordinates there are many unnecessary steps which may take some real time if the number of primitive internals is greater than 2000 or so (which is not so unusual). The timing problem may become acute in semiempirical calculations where the energy and gradient evaluations are cheap.

We briefly outline the theoretical background which is not difficult to understand:

Suppose, we have a set of n_I (redundant) primitive internal coordinates $\mathbf{q}$ constructed by some recipe and a set of $n_C = 3N_{\text{atoms}}$ Cartesian coordinates $\mathbf{x}$. The B-matrix is defined as:

$$B_{ij} = \frac{\partial q_i}{\partial x_j} \tag{4.1}$$

This matrix is rectangular. In order to compute the internal gradient one needs to compute the "generalized inverse" of $\mathbf{B}$. However, since the set of primitive internals is redundant the matrix is rank-deficient and one has to be careful. In practice one first computes the $n_I \times n_I$ matrix $\mathbf{G}$:

$$\mathbf{G} = \mathbf{B}\mathbf{B}^T \tag{4.2}$$

The generalized inverse of $\mathbf{G}$ is denoted $\mathbf{G}^-$ and is defined in terms of the eigenvalues and eigenvectors of $\mathbf{G}$:

$$\mathbf{G}^- = \begin{pmatrix} \mathbf{U} \\ \mathbf{R} \end{pmatrix}^T \begin{pmatrix} ?^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{U} \\ \mathbf{R} \end{pmatrix} \tag{4.3}$$

Here $\mathbf{U}$ are the eigenvectors belonging to the nonzero eigenvalues ? which span the nonredundant space and $\mathbf{R}$ are the eigenvectors of the redundant subspace of the primitive internal space. If the set of primitive internals is carefully chosen, then there are exactly $3N_{\text{atoms}} - 6(5)$ nonzero eigenvalues of $\mathbf{G}$. Using this matrix, the gradient in internal coordinates can be readily computed from the (known) Cartesian gradient:

$$\mathbf{g}_q = \mathbf{G}^-\mathbf{B}\mathbf{g}_x \tag{4.4}$$

The initial Hessian is formed directly in the redundant internal space and then itself or its inverse is updated during the geometry optimization.

Before generating the Newton step we have to ensure that the displacements take place only in the nonredundant part of the internal coordinate space. For this purpose a projector P':

$$\mathbf{P}' = \mathbf{G}\mathbf{G}^- = \mathbf{G}^-\mathbf{G} \tag{4.5}$$

is applied on both the gradient and the Hessian:

$$\mathbf{g_q} = \mathbf{P}'\mathbf{g_q} \tag{4.6}$$

$$\mathbf{H_q} = \mathbf{P}'\mathbf{H_q}\mathbf{P}' + \alpha\left(\mathbf{1} - \mathbf{P}'\right) \tag{4.7}$$

The second term for $\tilde{H}$ sets the matrix elements of the redundant part of the internal coordinate space to very large values ($\alpha = 1000$).

Coordinate steps

A Quasi-Newton (QN) step is the simplest choice to update the coordinates and is given by:

$$\Delta\mathbf{q} = -\mathbf{H}_\mathbf{q}^{-1}\mathbf{g}_\mathbf{q} \tag{4.8}$$

A more sophisticated step is the rational function optimization step which proceeds by diagonalizing the augmented Hessian:

$$\begin{pmatrix} \mathbf{H}_q & \mathbf{g}_q \\ \mathbf{g}_q & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{q} \\ 1 \end{pmatrix} = v \begin{pmatrix} \Delta\mathbf{q} \\ 1 \end{pmatrix} \tag{4.9}$$

The lowest eigenvalue ν_0 approaches zero as the equilibrium geometry is approached and the nice side effect of the optimization is a step size control. Towards convergence, the RFO step is approaching the quasi-Newton step and before it leads to a damped step is taken. In any case, each individual element of $\Delta\mathbf{q}$ is restricted to magnitude `MaxStep` and the total length of the step is restricted to `Trust`. In the RFO case, this is achieved by minimizing the predicted energy on the hypersphere of radius `Trust` which also modifies the direction of the step while in the quasi-Newton step, the step vector is simply scaled down.

Thus, the new geometry is given by:

$$\mathbf{q}_{\text{new}} = \mathbf{q}_{\text{old}} + \Delta\mathbf{q} \tag{4.10}$$

However, which Cartesian coordinates belong to the new redundant internal set? This is a somewhat complicated problem since the relation between internals and Cartesians is very nonlinear and the step in internal coordinates is not infinitesimal. Thus, an iterative procedure is taken to update the Cartesian coordinates. First of all consider the first (linear) step:

$$\Delta\mathbf{x} = \mathbf{A}\Delta\mathbf{q} \tag{4.11}$$

with $\mathbf{A} = \mathbf{B}^T\mathbf{G}^-$. With the new Cartesian coordinates $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$ a trial set of internals $\mathbf{q}_{k+1}$ can be computed. This new set should ideally coincide with $\mathbf{q}_{\text{new}}$ but in fact it usually will not. Thus, one can refine the Cartesian step by forming

$$\Delta\Delta\mathbf{q} = \mathbf{q}_{\text{new}} - \mathbf{q}_{k+1} \tag{4.12}$$

which should approach zero. This leads to a new set of Cartesians $\Delta\mathbf{x}' = \mathbf{A}\Delta\Delta\mathbf{q}$ which in turn leads to a new set of internals and the procedure is iterated until the Cartesians do not change and the output internals equal $\mathbf{q}_{new}$ within a given tolerance (10^{-7} RMS deviation in both quantities is imposed in ORCA).

Constrained Optimization

Constraints on the redundant internal coordinates can be imposed by modifying the above projector P' with a projector for the constraints C:

$$\mathbf{P} = \mathbf{P}' - \mathbf{P}'\mathbf{C}\left(\mathbf{CPC}\right)^{-1}\mathbf{CP}' \tag{4.13}$$

C is a diagonal matrix with 1's for the constraints and 0's elsewhere. The gradient and the Hessian are projected with the modified projector:

$$\tilde{g}_q = Pg_q \tag{4.14}$$

$$\tilde{H}_q = PH_qP + \alpha\left(1 - P\right) \tag{4.15}$$

Running Constrained Optimizations - examples

You can perform constrained optimizations which can, at times, be extremely helpful. This works as shown in the following example:

```
! RKS B3LYP/G SV(P) Opt
%geom Constraints
        { B 0 1 1.25 C }
        { A 2 0 3 120.0 C }
        end
      end

* int 0 1
    C  0 0 0 0.0000   0.000    0.00
    O  1 0 0 1.2500   0.000    0.00
    H  1 2 0 1.1075 122.016    0.00
    H  1 2 3 1.1075 122.016  180.00
*
```

```
Constraining bond distances        :   { B N1 N2 value C }
Constraining bond angles           :   { A N1 N2 N1 value C }
Constraining dihedral angles       :   { D N1 N2 N3 N4 value C }
Constraining cartesian coordinates :   { C N1 C }
Constraining only X-coordinates    :   { X N1 C }
Constraining only Y-coordinates    :   { Y N1 C }
Constraining only Z-coordinates    :   { Z N1 C }
```

Note

- Like for normal optimizations you can use numerical gradients (see *Numerical Gradients*.) for constrained optimizations. In this case the numerical gradient will be evaluated only for non-constrained coordinates, saving a lot of computational effort, if a large part of the structure is constrained.
- "value" in the constraint input is optional. If you do not give a value, the present value in the structure is constrained. For cartesian constraints you can't give a value, but always the initial position is constrained.
- It is recommended to use a value not too far away from your initial structure.
- It is possible to constrain whole sets of coordinates:

```
all bond lengths where N1 is involved       : { B N1 * C}
all bond lengths                            : { B * * C}
all bond angles where N2 is the central atom: { A * N2 * C }
all bond angles                             : { A * * * C }
all dihedral angles with central bond N2-N3 : { D * N2 N3 * C }
all dihedral angles                         : { D * * * * C }
```

- For Cartesian constraints (full or X/Y/Z-only) lists of atoms can be defined:

```
a list of atoms (10 to 17) with Cartesian constraints  : { C 10:17 C}
a list of atoms (0 to 5) with Cartesian Y constraints  : { Y 0:5 C}
```

- If there are only a few coordinates that have to be optimized you can use the `invertConstraints` option:

```
%geom Constraints
        { B 0 1 C }
        end
      invertConstraints true # only the C-O distance is optimized
                             # does not affect Cartesian coordinates
      end
```

- In some cases it is advantageous to optimize only the positions of the hydrogen atoms and let the remaining molecule skeleton fixed:

```
%geom optimizehydrogens true
      end
```

Note

- In the special case of a fragment optimization (see next point) the `optimizehydrogens` keyword does not fix the heteroatoms, but ensures that all hydrogen positions are relaxed.
- In Cartesian optimization, only Cartesian constraints are allowed.

Constrained Fragments and Molecular Cluster Optimization

The constrained fragments option was implemented in order to provide a convenient way to handle constraints for systems consisting of several molecules. The difference to a common optimization lies in the coordinate setup. In a common coordinate setup the internal coordinates are built up as described in the following:

In a first step, bonds are constructed between atom pairs which fulfill certain (atom type specific) distance criteria. If there are fragments in the system, which are not connected to each other (this is the case when there are two or more separate molecules), an additional bond is assigned to the nearest atom pair between the nonbonded fragments. All other internal coordinates are constructed on the basis of this set of bonds. Here, in a second step, bond angles are constructed between the atoms of directly neighboured bonds. If such an angle reaches more than 175°, a special type of linear angles is constructed. In a third step, dihedral angles (and improper torsions) are constructed between the atoms of directly neighbouring angles.

If the constrained fragments option is switched on, the set of bonds is constructed in a different way. The user defines a number of fragments (details on fragments definition can be found in *Fragment Specification* section.

For each fragment a full set of bonds (not seeing the atoms of the other fragments) is constructed as described above. When using this option, the user also has to define which fragments are to be connected. The connection between these fragments can either be user-defined or automatically chosen. If the user defines the connecting atoms N1 and N2, then the interfragmental bond is the one between N1 and N2. If the user does not define the interfragmental bond, it is constructed between the atom pair with nearest distance between the two fragments. Then the angles and dihedrals are constructed upon this (different) set of bonds in the already described fashion.

Now let us regard the definition of the fragment constraints: A fragment is constrained internally by constraining all internal coordinates that contain only atoms of the respective fragment. The connection between two fragments A and B is constrained by constraining specific internal coordinates that contain atoms of both fragments. For bonds, one atom has to belong to fragment A and the other atom has to belong to fragment B. Regarding angles, two atoms have to belong to fragment A and one to fragment B and vice versa. With respect to dihedrals, only those are constrained where two atoms belong to fragment A and the other two belong to fragment B.

Here is an example : if you want to study systems, which consist of several molecules (e.g. the active site of a protein) with constraints, then you can either use cartesian constraints (see above) or use ORCA's fragment constraint option. ORCA allows the user to define fragments in the system. For each fragment one can then choose separately whether it should be optimized or constrained. Furthermore, it is possible to choose fragment pairs whose distance and orientation with respect to each other should be constrained. Here, the user can either define the atoms which make up the connection between the fragments, or the program chooses the atom pair automatically via a closest distance criterion. ORCA then chooses the respective constrained coordinates automatically. An example for this procedure is shown below.

The coordinates are taken from a crystal structure [PDB-code 2FRJ]. In our gas phase model we choose only a small part of the protein, which is important for its spectroscopic properties. Our selection consists of a heme-group (fragment 1), important residues around the reaction site (lysine (fragment 2) and histidine (fragment 3)), an important water molecule (fragment 4), the NO-ligand (fragment 5) and part of a histidine (fragment 6) coordinated to the heme-iron. In this constrained optimization we want to maintain the position of the heteroatoms of the heme group. Since the protein backbone is missing, we have to constrain the orientation of lysine and histidine (fragments 2 and 3) side chains to the heme group. All other fragments (the ones which are directly bound to the heme-iron and the water molecule) are fully optimized internally and with respect to the other fragments. Since the crystal structure does not reliably resolve the hydrogen positions, we relax also the hydrogen positions of the heme group.

```
# !! If you want to run this optimization: be aware
# !!  that it will take some time!
! BP86 SV(P) Opt
%geom
    ConstrainFragments { 1 } end # constrain all internal
                                 # coordinates of fragment 1
    ConnectFragments
     {1 2 C 12 28}  # connect the fragments via the atom pair 12/28 and 15/28 and
     {1 3 C 15 28}  #  constrain the internal coordinates connecting
                    #  fragments 1/2 and 1/3
      {1 5 O}
      {1 6 O}
      {2 4 O}
      {3 4 O}
    end
    optimizeHydrogens true  # do not constrain any hydrogen position
  end
* xyz 1 2
Fe(1)     -0.847213     -1.548312     -1.216237 newgto "TZVP" end
```

(continues on next page)

(continued from previous page)

```
N(5)     -0.712253     -2.291076      0.352054 newgto "TZVP" end
O(5)     -0.521243     -3.342329      0.855804 newgto "TZVP" end
N(6)     -0.953604     -0.686422     -3.215231 newgto "TZVP" end
N(3)     -0.338154     -0.678533      3.030265 newgto "TZVP" end
N(3)     -0.868050      0.768738      4.605152 newgto "TZVP" end
N(6)     -1.770675      0.099480     -5.112455 newgto "TZVP" end
N(1)     -2.216029     -0.133298     -0.614782 newgto "TZVP" end
N(1)     -2.371465     -2.775999     -1.706931 newgto "TZVP" end
N(1)      0.489683     -2.865714     -1.944343 newgto "TZVP" end
N(1)      0.690468     -0.243375     -0.860813 newgto "TZVP" end
N(2)      1.284320      3.558259      6.254287
C(2)      5.049207      2.620412      6.377683
C(2)      3.776069      3.471320      6.499073
C(2)      2.526618      2.691959      6.084652
C(3)     -0.599599     -0.564699      6.760567
C(3)     -0.526122     -0.400630      5.274274
C(3)     -0.194880     -1.277967      4.253789
C(3)     -0.746348      0.566081      3.234394
C(6)      0.292699      0.510431     -6.539061
C(6)     -0.388964      0.079551     -5.279555
C(6)      0.092848     -0.416283     -4.078708
C(6)     -2.067764     -0.368729     -3.863111
C(1)     -0.663232      1.693332     -0.100834
C(1)     -4.293109     -1.414165     -0.956846
C(1)     -1.066190     -4.647587     -2.644424
C(1)      2.597468     -1.667470     -1.451465
C(1)     -1.953033      1.169088     -0.235289
C(1)     -3.187993      1.886468      0.015415
C(1)     -4.209406      0.988964     -0.187584
C(1)     -3.589675     -0.259849     -0.590758
C(1)     -3.721903     -2.580894     -1.476315
C(1)     -4.480120     -3.742821     -1.900939
C(1)     -3.573258     -4.645939     -2.395341
C(1)     -2.264047     -4.035699     -2.263491
C(1)      0.211734     -4.103525     -2.488426
C(1)      1.439292     -4.787113     -2.850669
C(1)      2.470808     -3.954284     -2.499593
C(1)      1.869913     -2.761303     -1.932055
C(1)      2.037681     -0.489452     -0.943105
C(1)      2.779195      0.652885     -0.459645
C(1)      1.856237      1.597800     -0.084165
C(1)      0.535175      1.024425     -0.348298
O(4)     -1.208602      2.657534      6.962748
H(3)     -0.347830     -1.611062      7.033565
H(3)     -1.627274     -0.387020      7.166806
H(3)      0.121698      0.079621      7.324626
H(3)      0.134234     -2.323398      4.336203
H(3)     -1.286646      1.590976      5.066768
H(3)     -0.990234      1.312025      2.466155
H(4)     -2.043444      3.171674      7.047572
H(2)      1.364935      4.120133      7.126900
H(2)      0.354760      3.035674      6.348933
H(2)      1.194590      4.240746      5.475280
H(2)      2.545448      2.356268      5.027434
H(2)      2.371622      1.797317      6.723020
H(2)      3.874443      4.385720      5.867972
H(2)      3.657837      3.815973      7.554224
H(2)      5.217429      2.283681      5.331496
H(2)      5.001815      1.718797      7.026903
H(6)     -3.086380     -0.461543     -3.469767
H(6)     -2.456569      0.406212     -5.813597
```

(continues on next page)

(continued from previous page)

```
 H(6)      1.132150     -0.595619     -3.782287
 H(6)      0.040799      1.559730     -6.816417
 H(6)      0.026444     -0.139572     -7.404408
 H(6)      1.392925      0.454387     -6.407850
 H(1)      2.033677      2.608809      0.310182
 H(1)      3.875944      0.716790     -0.424466
 H(1)      3.695978     -1.736841     -1.485681
 H(1)      3.551716     -4.118236     -2.608239
 H(1)      1.487995     -5.784645     -3.308145
 H(1)     -1.133703     -5.654603     -3.084826
 H(1)     -3.758074     -5.644867     -2.813441
 H(1)     -5.572112     -3.838210     -1.826943
 H(1)     -0.580615      2.741869      0.231737
 H(1)     -3.255623      2.942818      0.312508
 H(1)     -5.292444      1.151326     -0.096157
 H(1)     -5.390011     -1.391441     -0.858996
 H(4)     -1.370815      1.780473      7.384747
 H(2)      5.936602      3.211249      6.686961
*
```

Note

- You have to connect the fragments in such a way that the whole system is connected.
- You can divide a molecule into several fragments.
- Since the initial Hessian for the optimization is based upon the internal coordinates: Connect the fragments in a way that their real interaction is reflected.
- This option can be combined with the definition of constraints, scan coordinates(see *Surface Scans*) and the `optimizeHydrogens` option (but: its meaning in this context is different to its meaning in a normal optimization run, relatively straightforward see section *Geometry Optimizations*).
- Can be helpful in the location of complicated transition states (with relaxed surface scans), see *Surface Scans*.
- Since constraints must be defined manually, *Automatic Fragmentation* will not be enabled automatically if there are unassigned atoms to fragments. However, it can be activated by including a `%frag` block.

4.1.4 Optimization Assuming a Rigid Body

If you want to optimize one or multiple fragments assuming these as a "rigid bodies", i.e. only allowing for movements of the center of mass and pure rotations for each fragment, the keyword `!RIGIDBODYOPT` can be used.

It needs to be combined with one of the optimization flags, such as `!OPT RIGIDBODYOPT` or `!L-OPT RIGIDBODYOPT`, and all fragments which were defined in any way will be treated as such.

For a more fine control, whenever fragments are defined for the system, each fragment can be optimized differently (similar to the fragment optimization described above). The following options are available: `FixFrags`,`RelaxHFrags`,`RelaxFrags`, and `RigidFrags` shown within Section *Geometry Optimization Keywords*.

A more complex example is depicted in the following:

```
! L-Opt # or !COPT or !OPT

*xyzfile 0 1 geometry.xyz

%frag
 Definition
```

(continues on next page)

(continued from previous page)

```
  1 {0:12} end
  2 {13:17} end
  3 {18:38} end
  4 {39:53} end
 end
end

%geom
 RelaxFrags {1} end     # Fragment 1 is fully relaxed
 RigidFrags {2 3 4} end # Fragments 2, 3 and 4 are treated as rigid bodies each.
end
```

Note

The manual definition of fragments have to be done after the coordinate input.

Important

- If there are atoms not assigned to any fragment, *Automatic Fragmentation* will be enabled by default. Users can explicitly control this behavior, as described in *Fragment Specification*.
- Fragment enumeration will be reordered if manual assignments exceed the total number of fragments and *Automatic Fragmentation* is enabled. However, the values of `RelaxFrags` and `RigidFrags` will not be updated accordingly.

4.1.5 Adding Arbitrary Bias and Wall Potentials

For some applications, it might be interesting to add arbitrary bias potentials during the geometry optimization. For example, if you want to optimize an intermolecular complex and need that both structures stick together, without one flying away during the optimization, you might want to add a wall bias. Or you might need a bond bias to guide the DOCKER to dock structures onto certain specific regions of others.

Note

All these walls and biases are integrated with the other electronic-structure methods in ORCA, including their Hessian.

Bond Bias Potentials

From ORCA 6.1, we introduced a bond bias potential that can be added between two arbitrary atoms. It will effectively create a fictitious bond between atoms A and B, using a symmetric Morse-like potential:

$$V = E_{AB}(1 - e^{-\alpha(R_{AB}-R_{min})^2})$$

where E_{AB} is the energy difference from the bottom of the well to the top, R_{min} is the equilibrium distance. α is a decay parameter that will be automatically determined if not given such as to ensure 50% intensity at $2 \times R_{min}$. In practice, it looks like this:

The input to create a simple bias between atoms 0 and 1 is:

```
%GEOM
  BIAS { B 0 1 } END
END
```

Up to three other parameters can be given: R_{min} in Angstroem, E_{AB} in kcal/mol and the alpha. If none of those are given the default values are: sum of Van der Walls radii, 25 kcal/mol and α will be chosen as described above.

In order to define a bond with $R_{min} = 1.5\AA$ and a $E_{AB} = 100kcal/mol$, for example:

```
%GEOM
  BIAS { B 0 1 1.5 100 } END
END
```

If a star symbol (*) is used for any of those three, that quantity will be taken from the defaults. For example, in order to set the energy depth, but not the distance, simply use:

```
%GEOM
  BIAS { B 0 1 * 100 } END
END
```

Note

A negative number on the E_{AB} will create the inverse, repulsive potential instead!

Important

The ORCA DOCKER has its own `BIAS` block and needs a different input than this to work there, please check Section *Adding a bond bias to the docking process* for more details.

Wall Bias Potentials

you can also add three kinds of arbitrary "wall potentials": an ellipsoid or spherical of the form

$$V = \left(\frac{|\mathbf{R} - \mathbf{O}|}{radius}\right)^{30}$$

or a rectangular box potential with 6 walls of the form

$$V = e^{5(\mathbf{R} - wall)}$$

These can be given in two ways: by explicitly defining the origin of the potential and its limits, e.g:

```
%GEOM
    ELLIPSEPOT 0,0,0,5,3,4 # the last are the a,b and c radii
END
```

or:

```
%GEOM
    SPHEREPOT 0,0,0,5 # the last is the radius
END
```

or:

```
%GEOM
    BOXPOT 0,0,0,4,-4,3,-3,6,-6 # maxx, minx, maxy, miny, maxz and minz last
END
```

where the first three numbers are the center and the last is the radius for the sphere (or a,b and c for the ellipsoid) and the max and min x,y and z dimensions of the box. All numbers should be given in Ångström.

In case a single number is given instead, the walls will be automatically centered around the centroid of the molecule and that number will be added to the minimum sphere or box that is necessary to contain the molecule. For example:

```
%GEOM
    SPHEREPOT 2
END
```

or:

```
%GEOM
    BOXPOT 2
END
```

will build a minimum wall centered on the centroid that encloses the molecule and add 2 Ångström on top of it. Still on the sphere case, a negative number like

```
%GEOM
    SPHEREPOT -2
END
```

will make the total radius of the sphere to be Ångström.

Note

This will apply to regular geometry optimizations, as well as to the Global Optimizer (GOAT), NEB, DOCKER, SOLVATOR, etc.

4.1.6 Constant External Force - Mechanochemistry

Constant external force can be applied on the molecule within the EFEI formalism[539] by pulling on the two defined atoms. To apply the external force, use the POTENTIALS in the geom block. The potential type is C for Constant force, indexes of two atoms (zero-based) and the value of force in nN.

```
! def2-svp  OPT
%geom
  POTENTIALS
    { C 2 3 4.0 }
  end
end

 * xyz 0 1
O          0.73020       -0.07940       -0.00000
O         -0.73020        0.07940       -0.00000
H          1.21670        0.75630        0.00000
H         -1.21670       -0.75630        0.00000
 *
```

The results are seen in the output of the SCF procedure, where the total energy already contains the force term.

```
----------------
TOTAL SCF ENERGY
----------------

Total Energy       :         -150.89704913 Eh           -4106.11746 eV

Components:
Nuclear Repulsion  :           36.90074715 Eh            1004.12038 eV
External potential :           -0.25613618 Eh              -6.96982 eV
Electronic Energy  :         -187.54166010 Eh           -5103.26802 eV
:language: orca
```

4.1.7 Printing Hessian in Internal Coordinates

When a Hessian is available, it can be printed out in redundant internal coordinates as in the following example:

```
! opt
%geom inhess read
      inhessname "h2o.hess"
      PrintInternalHess true
end
*xyz 0 1
  O      0.000000    0.000000    0.000000
  H      0.968700    0.000000    0.000000
  H     -0.233013    0.940258    0.000000
*
```

Note

- The Hessian in internal coordinates is (for the input `printHess.inp`) stored in the file `printHess_internal.hess`.
- The corresponding lists of redundant internals is stored in `printHess.opt`.
- Although the `!Opt` keyword is necessary, an optimization is not carried out. ORCA exits after storing the Hessian in internal coordinates.

4.1.8 Using model Hessian from previous calculations

If you had a geometry optimization interrupted, or for some reason want to use the model Hessian updated from a previous calculation, you can do that by passing a `basename.opt` file, a `basename.carthess` file or the initial Hessian on a new calculation.

```
%GEOM
    InHess     READ
    InHessName "basename.carthess"
END
```

4.1.9 Geometry Optimizations using the L-BFGS optimizer

Optimizations using the L-BFGS optimizer are done in Cartesian coordinates. They can be invoked quite simple as in the following example:

```
! L-Opt
! MM
%MM
  ORCAFFFILENAME "CHMH.ORCAFF.prms"
END
*pdbfile 0 1 CHMH.pdb
```

Using this optimizer systems with 100s of thousands of atoms can be optimized. Of course, the energy and gradient calculations should not become the bottleneck for such calculations, thus MM or QM/MM methods should be used for such large systems.

The default maximum number of iterations is 200, and can be increased as follows:

```
! L-Opt
%GEOM
  maxIter 500 # default 200
END
*pdbfile 0 1 CHMH.pdb
```

Only the hydrogen positions can be optimized with the following command:

```
! L-OptH
```

But also other elements can be exclusively optimized with the following command:

```
! L-OptH
%GEOM
 OptElement F  # optimize fluorine only when L-OptH is invoked.
               # Does not work with optimization in internal coordinates.
END
```

Note

If you want to use the previously (ORCA5 and ORCA6.0) available `L-Opt` option, you need to specify `MD-L-Opt` or `MD-L-OptH`.

4.1.10 Optimizing with External Methods

Although ORCA features a many electronic structure models, it cannot cover all available methods. Therefore, ORCA features an interface to external programs that allows you to use ORCA's features with electronic structure methods and model potentials provided by external programs. This way, you'll be able to utilize the latest MNDO based semi-empirical methods like PM7 or neural network potentials like AIMNet2 and UMA within the ORCA infrastructure, e.g. to *optimize structures*, search transition states with *NEB-TS*, or sample conformers with *GOAT*.

The external optimization infrastructure can be accessed via the ORCA simple input keyword `!ExtOpt` which can be combined with other common keywords like `!Opt` or `!GOAT`. As every program typically provides differently formatted output, individual wrapper scripts are required to link ORCA and its external optimizer infrastructure. The full path to your wrapper script can be provided via the `%method` block in the ORCA input file:

```
! ExtOpt

%method
  ProgExt "/full/path/to/wrapperscript"
  Ext_Params "optional command line arguments"
end
```

Alternatively, you can also provide the script as a file or link named `otool_external` in the same directory as the ORCA executables, or by assigning the `EXTOPTEXE` environment variable to the *full path* to the external program.

Regardless of which option is used, the keyword `Ext_Params` can be used to specify the additional command line arguments as a single string.

> **Note**
>
> All information that you give on the electronic structure is discarded.

After executing ORCA, it automatically writes an input file called `basename_EXT.extinp.tmp` in each step containing the following info:

```
basename_EXT.xyz # xyz filename: string, ending in '.xyz'
0 # charge: integer
1 # multiplicity: positive integer
1 # NCores: positive integer
0 # do gradient: 0 or 1
pointcharges.pc # point charge filename: string (optional)
```

Comments from # until the end of the line should be ignored. The file `basename_EXT.xyz` will also be written to the working directory. While the energy is always requested, the gradient is only calculated when needed.

ORCA then calls:

```
scriptname basename_EXT.extinp.tmp [args]
```

Here, `args` are the optional command line arguments provided defined with the `Ext_Params` in the ORCA input. They are passed to the external program without any changes.

The external script starts the energy (and gradient) calculation and finally provides the results in a file called `basename_EXT.engrad` using the same `basename` as the XYZ file. This file must have the following format:

```
#
# Number of atoms: must match the XYZ
#
3
#
# The current total energy in Eh
#
```

(continues on next page)

(continued from previous page)

```
-5.504066223730
#
# The current gradient in Eh/bohr: Atom1X, Atom1Y, Atom1Z, Atom2X, etc.
#
-0.000123241583
0.000000000160
-0.000000000160
0.000215247283
-0.000000001861
0.000000001861
-0.000092005700
0.000000001701
-0.000000001701
```

Comments from # until the end of the line are ignored, as are any comment-only lines.

The script may also print relevant output to `STDOUT` and/or `STDERR`. `STDOUT` will either be printed in the ORCA standard output, or redirected to a temporary file and removed afterwards, depending on the type of job (e.g., for `NumFreq` the output for the displaced geometries is always redirected) and the ORCA output settings:

```
%output
  Print[P_EXT_OUT]  1 # (default) print the external program output
  Print[P_EXT_GRAD] 1 # (default) print the gradient from the ext. program
end
```

Important

Some wrapper scripts and how to use them can be found in the official GitHub repository. While some methods and programs such as MOPAC, AIMNet2, and UMA are already supported, you are free to write your own wrapper scripts to extend the available methods usable with ORCA. If you do so, please contribute to the repository via pull requests to share with the community.

Note

Note that depending on your OS and infrastructure, you may need to do some modifications to the wrapper script.

4.1.11 Optimization FAQs

I can't locate the transition state (TS) for a reaction expected to feature a low/very low barrier, what should I do?

For such critical case of locating the TS, running a very fine (e.g. 0.01 Å increment of the bond length) relaxed scan of the key reaction coordinate is recommended, see *Surface Scans*. In this way the highest energy point on a very shallow surface can be identified and used for the final TS optimisation.

During the geometry optimisation some atoms merge into each other and the optimisation fails. How can this problem be solved?

This usually occurs due to the wrong or poor construction of initial molecular orbital involving some atoms. Check the basis set definition on problematic atoms and then the corresponding MOs!

4.1.12 Geometry Optimization Keywords

Table 4.1: Simple input keywords for geometry optimization.

Keyword	Description
`Opt`	Optimization with default convergence tolerance (Alias: `NormalOpt`)
`LooseOpt`	Optimization with looser convergence tolerance
`TightOpt`	Optimization with tight convergence tolerance
`VeryTightOpt`	Optimization with very tight convergence tolerance
`COpt`	Optimization in Cartesian coordinates (can be combined with `TightOpt`, etc.)

Table 4.2: `%method` block input keywords and options for geometry optimization.

Keyword	Option / Value	Description
`%method`	`RunTyp Opt`	Sets geometry optimization as the run type.
	`RunTyp=Geom`	Alternative syntax for geometry optimization.

Table 4.3: `%geom` block input keyword

Keyword	Option / Value	Description
`MaxIter`	`50`	Maximum number of geometry iterations (default max(3N, 50)).
`coordsys`	`redundant`	Use redundant internal coordinates.
	`cartesian`	Use Cartesian coordinates.
`cartfallback`	`true`	Use Cartesian step if internal coordinate step fails.
`modify_internal`	`{B 10 0 A}`	Add bond between atoms 0 and 10.
	`{A 8 9 10 R}`	Remove angle between atoms 8, 9, and 10.
	`{D 7 8 9 10 R}`	Remove dihedral between atoms 7, 8, 9, and 10.
`Constraints`	`{B 1 2 value C}`	Constrain bond between atoms 1 and N.
	`{A 1 2 3 value C}`	Constrain angle defined by atoms 1, 2, and 3.
	`{D 1 2 3 4 value C}`	Constrain dihedral defined by atoms 1–4.
	`{C 1 C}`	Constrain Cartesian position of atom 1.
	`{B 1 * C}`	Constrain all bonds involving atom 1.
	`{B * * C}`	Constrain all bonds.
	`{A * 2 * C}`	Constrain all angles with atom 2 as central.
	`{A * * * C}`	Constrain all angles.
	`{D * 2 3 * C}`	Constrain all dihedrals with 2 and 3 as central atoms.
	`{D * * * * C}`	Constrain all dihedrals.
`ReduceRedInts`	`true`	Generates the internal coordinates of the constrains atoms.
`ConnectFragments`	`{1 3 O}`	Optimize internal coordinates connecting fragments 1 and 3.
	`{1 3 O N1 N2}`	Connect fragments 1 and 3 via atoms N1 and N2.
`ConstrainFragments`	`{1}`	Constrain all internal coordinates within fragment 1.
`Frags`	`X {0:12}`	Defining fragment X, containing atoms 0-12. Note that all other atoms not specified in th
`AUTOFRAG`	`true`	Automatically select fragments based on topology
`FixFrags`	`{1}`	Freeze the coordinates of all atoms of the specified fragment(s).
`RelaxHFrags`	`{1}`	Relax the hydrogen atoms of the specified fragment(s). Default for all atoms if !L-OptH
`RelaxFrags`	`{1}`	Relax all atoms of the specified fragment(s). Default for all atoms if !L-Opt is defined.
`RigidFrags`	`{1}`	Treat each specified fragment as a rigid body, but relax the position and orientation of th
`optimizeHydrogens`	`true`	Optimize coordinates involving H atoms while constraining others.
`freezeHydrogens`	`true`	Freeze H positions relative to heteroatoms.
`invertConstraints`	`true`	Invert the defined constraints (only affects redundant internals).
`Step`	`qn`	Use quasi-Newton step.
	`rfo`	Use Rational Function Optimization step (default for `!Opt`).
	`prfo`	Use partitioned RFO step (default for `!OptTS`).
`MaxStep`	`0.3`	Maximum step length in internal coordinates (default: 0.3 au).
`Trust`	`-0.3`	Use fixed trust radius of 0.3 au (negative sign indicates fixed).
`TolE`	`5e-6`	Convergence threshold for energy change (a.u.).
`TolRMSG`	`1e-4`	Convergence threshold for RMS gradient (a.u.).
`TolMaxG`	`3e-4`	Convergence threshold for max gradient component (a.u.).

Table 4.3 – continue

Keyword	Option / Value	Description
TolRMSD	2e-3	Convergence threshold for RMS displacement (a.u.).
TolMaxD	4e-3	Convergence threshold for max displacement (a.u.).
Convergence	normal	Default convergence threshold.
	loose	Looser convergence thresholds.
	tight	Tighter convergence thresholds.
ProjectTR	false	Disable translation and rotation projection (must be false for internals).
inhess	unit	Start optimization with a unit matrix Hessian.
	Read	Read Hessian from external file. Requires InHessName.
InHessName	"filename.hess"	Specifies file containing the Hessian (e.g., .hess, .opt, .carthess).
	Lindh	Use Lindh's model Hessian (only for redundant coordinates).
	Almloef	Use Almloef's model Hessian.
	Schlegel	Use Schlegel's model Hessian.
	Swart	Use Swart and Bickelhaupt's model Hessian.
	XTB0	Use GFN0-xTB Hessian.
	XTB1	Use GFN1-xTB Hessian.
	XTB2	Use GFN2-xTB Hessian.
	GFNFF	Use GFN-FF Hessian.
Calc_Hess	true	Calculate numerical Hessian at the beginning of optimization.
Recalc_Hess	5	Recalculate Hessian every 5 cycles.
Hybrid_Hess	{0 1 5 6} end	Calculate exact Hessian for specified atoms, use model for the rest.
NumHess	true	Request numerical Hessian.
Hess_Internal	{A 3 2 1 D 2.0}	Manually set Hessian value (Eh/Bohr^2) for angle between atoms 3-2-1.
	reset 5	Reset modified Hessian values after 5 cycles.
	{B 1 0 C}	Coordinates for Hessian refinement after a relaxed surface scan.
	XYZ1 "scanName.003.xyz"	Structure file before maximum in scan.
	XYZ2 "ScanName.005.xyz"	Structure file after maximum in scan.
	GBW1 "ScanName.003.gbw"	Optional GBW file before max in scan.
	GBW2 "ScanName.005.xyz"	Optional GBW file after max in scan.
Update	Powell	Use Powell update for Hessian.
	Bofill	Use Bofill update (default for TS optimization).
	BFGS	Use BFGS update (default for geometry optimization).
HESS_Modification	Shift_Diag	Shift diagonal elements of Hessian (default).
	EV_Reverse	Reverse sign of diagonal elements.
HESS_MinEV	0.0001	Minimum allowed Hessian eigenvalue.
NResetHess	20	Rebuild model Hessian after 20 steps (BFGS only).
NStepsInResetHess	5	Number of previous steps used to inform new model Hessian.
UseSOSCF	false	Switches the converger to SOSCF after the first point. May converge better than DIIS if
ReducePrint	true	Reduces printout after the first point. Default: true.
OptGuess		The initial guess can be changed after the first point. The default is MORead. The MOs
OptGuess	OneElec	Use the one-electron matrix guess.
	Hueckel	Use the extended Hückel guess.
	PAtom	Use the PAtom guess.
	PModel	Use the PModel guess.
	MORead	Use MOs of the previous point. Default.
AddExtraBonds	true	Enables assigning bonds to atom pairs connected by more than MaxLength bonds and
AddExtraBonds_MaxLength	10	Maximum number of bonds connecting two atoms to consider for extra bonding. Defaul
AddExtraBonds_MaxDist	5	Maximum distance (in Å) between two atoms to be considered for extra bonding. Defau
BIAS	{B 0 1}	Create a fictitious bond between atoms 1 and 2 to create Morse-like potential
ELLIPSEPOT	0,0,0,5,3,4	Defines an ellipsoidal potential centered at (0,0,0) with semi-axes a=5, b=3, c=4.
SPHEREPOT	0,0,0,5	Defines a spherical potential centered at (0,0,0) with radius 5.
BOXPOT	0,0,0,4,-4,3,-3,6,-6	Defines a box potential: maxx=4, minx=–4, maxy=3, miny=–3, maxz=6, minz=–6.
OptElement	N	Optimize element "N" atoms only (requires L-OptH; not compatible with internal coordi

4.2 Surface Scans

Relaxed surface scans are an invaluable tool when exploring potential energy surfaces. They allow the user to vary one or more coordinates (bond lengths, angles, or dihedrals) while relaxing the remaining degrees of freedom. This section outlines the theoretical background, gives practical examples, and summarizes relevant input keywords.

4.2.1 Theory

In a relaxed surface scan, a selected coordinate is varied over a defined range, and at each point, a constrained geometry optimization is performed. This can reveal transition states, reaction mechanisms, or approximate energy barriers. ORCA supports scans over bond lengths (`B`), bond angles (`A`), and dihedral angles (`D`).

Multiple coordinates can also be scanned either independently (nested loops) or simultaneously using the `Simul_Scan` keyword, the latter reducing computational cost.

4.2.2 Simple Example

The following input scans the bond length between C and O from 1.35 Å to 1.10 Å in 12 equidistant steps. Each step involves a constrained geometry optimization.

```
! B3LYP SV(P) Opt
%geom
     scan
        B 0 1 = 1.35, 1.10, 12 # C-O distance that will be scanned
     end
end

* int 0 1
    C  0 0 0 0.0000   0.000    0.00
    O  1 0 0 1.3500   0.000    0.00
    H  1 2 0 1.1075 122.016    0.00
    H  1 2 3 1.1075 122.016  180.00
*
```

Tip

- As in constrained geometry optimizations it is possible to start the relaxed surface scan with a different scan parameter than the value present in your molecule. But keep in mind that this value should not be too far away from your initial structure.

4.2.3 Complex Example

A more challenging example is shown below. Here, the H-atom abstraction step from CH_4 to OH-radical is computed with a relaxed surface scan (*vide supra*). The job was run as follows:

```
! B3LYP SV(P) Opt SlowConv NoTRAH
%geom
    scan
        B 1 0 = 2.0, 1.0, 15
    end
end
```

(continues on next page)

(continued from previous page)

```
* int 0 2
 C     0   0   0   0.000000     0.000     0.000
 H     1   0   0   1.999962     0.000     0.000
 H     1   2   0   1.095870   100.445     0.000
 H     1   2   3   1.095971    90.180   119.467
 H     1   2   3   1.095530    95.161   238.880
 O     2   1   3   0.984205   164.404    27.073
 H     6   2   1   0.972562   103.807    10.843
*
```

It is obvious that the reaction is exothermic and passes through an early transition state in which the hydrogen jumps from the carbon to the oxygen. The structure at the maximum of the curve is probably a very good guess for the true transition state that might be located by a transition state finder.

You will probably find that such relaxed surface scans are incredibly useful but also time consuming. Even the simple job shown below required several hundred single point and gradient evaluations (convergence problems appear for the SCF close to the transition state and for the geometry once the reaction partners actually dissociate – this is to be expected). Yet, when you search for a transition state or you want to get insight into the shapes of the potential energy surfaces involved in a reaction it might be a good idea to use this feature. One possibility to ease the burden somewhat is to perform the relaxed surface scan with a "fast" method and a smaller basis set and then do single point calculations on all optimized geometries with a larger basis set and/or higher level of theory. At least you can hope that this should give a reasonable approximation to the desired surface at the higher level of theory – this is the case if the geometries at the lower level are reasonable.

Fig. 4.1: Relaxed surface scan for the H-atom abstraction from CH_4 by OH-radical (B3LYP/SV(P)).

4.2.4 Multidimensional Scans

After several requests from our users ORCA now allows up to three coordinates to be scanned within one calculation.

```
! B3LYP SV(P) Opt
%geom
     scan
        B 0 1 = 1.35, 1.10, 12 # C-O distance that will be scanned
        B 0 2 = 1.20, 1.00, 5  # C-H distance that will be scanned
        A 2 0 1 = 140, 100, 5  # H-C-O angle that will be scanned
     end
end

* int 0 1
    C  0 0 0 0.0000   0.000    0.00
    O  1 0 0 1.3500   0.000    0.00
    H  1 2 0 1.1075 122.016    0.00
    H  1 2 3 1.1075 122.016  180.00
*
```

Note

- For finding transition state structures of more complicated reaction paths ORCA now offers its very efficient NEB-TS implementation (see section *Nudged Elastic Band Method*).
- 2-dimensional or even 3-dimensional relaxed surface scans can become very expensive - e.g. requesting 10 steps per scan, ORCA has to carry out 1000 constrained optimizations for a 3-D scan.
- The results can depend on the direction of the individual scans and the ordering of the scans.

Simultaneous multidimensional scans, in which all scan coordinates are changed at the same time, can be requested via the following keyword (which brings the cost of a multidimensional relaxed surface scan down to the cost of a single relaxed surface scan):

```
%geom
  Scan B 0 1 = 3, 1, 15 end
  Scan B 1 2 = 1, 3, 15 end
  Simul_Scan true
end
```

4.2.5 Multiple XYZ File Scans

A different type of scan is implemented in ORCA in conjunction with relaxed surface scans. Such scans produce a series of structures that are typically calculated using some ground state method. Afterwards one may want to do additional or different calculations along the generated pathway such as excited state calculations or special property calculations. In this instance, the "multiple XYZ scan" feature is useful. If you request reading from a XYZ file via:

```
* xyzfile Charge Multiplicity FileName
```

this file could contain a number of structures. The format of the file is:

```
Number of atoms M
  Comment line
AtomName1 X Y Z
AtomName2 X Y Z
...
AtomNameM X Y Z
Number of atoms M
  Comment line
```

(continues on next page)

(continued from previous page)

```
AtomName1 X Y Z
...
```

Thus, the structures are simply of the standard XYZ format and they are provided one after the other. There is no need to add any extra character between them. This was the case for ORCA versions older than 6.0.0, where the structures were separated by a ">" sign. The user can still use this format, if preferred, but is not needed anymore. After processing the XYZ file, single point calculations are performed on each structure in sequence and the results are collected at the end of the run in the same kind of trajectory `.dat` files as produced from trajectory calculations.

In order to aid in using this feature, the relaxed surface scans produce a file called `MyJob.allxyz` that is of the correct format to be re-read in a subsequent run.

4.2.6 Surface Scan Keywords

Table 4.4: `%geom` block input keywords and options for surface scanned geometry optimization.

Keyword	Option / Value	Description
`Scan`	`B 1 2 = v1, v2, N`	Scan atom 1 to atom 2 bond distance from v1 to v2 in N steps.
	`B 1 2 [v1 v2 ... ]`	Scan atom 1 to atom 2 bond with custom step values.
	`A 1 2 3`	Scan an angle using atoms 1,2,3.
	`D 1 2 3 4`	Scan a dihedral of atoms 1,2,3,4.
`Simul_Scan`	`true`	Enable simultaneous multidimensional scans (up to 3 coordinates).
`FullScan`	`true`	Forces full surface scan before TS optimization (`!ScanTS`).

4.3 Transition State Searches

4.3.1 Transition State Search Theory

As transition state finder we implemented the well-established eigenvector following algorithm using a P-RFO step as implemented by Baker [538]. This algorithm is a quasi-Newton like algorithm.

The Taylor series of the energy, truncated after the quadratic term, is:

$$E = E_0 + g_q{}^{+}\Delta q_q + \frac{1}{2}\Delta q{}^{+}\mathrm{H}_q\Delta q \tag{4.16}$$

The Newton-Raphson step to get from the actual point to a stationary point is:

$$\Delta q = -\mathrm{H}_q^{-1} g_q = \sum -\frac{V_i^{+} g_q V_i}{b_i} \tag{4.17}$$

with V_i and b_i as eigenvectors and eigenvalues of the Hessian H_q. This step leads to the nearest stationary point on the PES. This stationary point can be a minimum or a saddle point, according to the curvature of the PES at the actual point.

With a simple shift of the Hessian eigenvalues b_i in this equation one can guide the step to a stationary point with the required characteristics (Hessian with exactly one negative eigenvalue). The transition state search is separated into two different optimization problems. The energy is maximized along one Hessian eigenmode and minimized along the remaining $3N - 7(6)$ eigenmodes. We introduce two different shift parameters λ_p and λ_n, where λ_p is the shift parameter for the eigenmode being maximized and λ_n shifts the Hessian eigenvalues of the modes being minimized.

This method allows us to maximize along any mode, not only the one with smallest eigenvalue. Starting from two different RFO-matrices for the different optimization problems (see description above) we get for λ_p and λ_n:

$$\lambda_p = \frac{1}{2}b_k + \frac{1}{2}\sqrt{b_k^2 + 4F_k^2} \quad \text{and} \quad \sum_{i \neq k} \frac{F_i^2}{\lambda_n - b_i} = \lambda_n \tag{4.18}$$

whereas $F_i = V_i^+ g$ is the component of g along the Hessian eigenmode V_i and λ_n has to get solved iteratively. The solution for λ_n has to be negative and lower than b_2 (or lower than b_1, if not the lowest mode is being followed). If the Hessian has more than one negative eigenvalue, these properties might not be fulfilled, and the Hessian would have to be modified. In our implementation the Hessian diagonal elements are either shifted or reversed in such a case.

Once the shift parameters are known the P-RFO step h is calculated as follows:

$$\Delta q_k = -\frac{\bar{F}_k V_k}{b_k - \lambda_p} \quad \text{and} \quad \Delta q_i = -\frac{\bar{F}_i V_i}{b_i - \lambda_n} \quad \text{with} \quad i = 1 \ldots n, \;\; i \neq k \tag{4.19}$$

$$\Delta q = \sum_{j=1}^{n} \Delta q_j \tag{4.20}$$

4.3.2 Introduction to Transition State Searches

If you provide a good estimate for the structure of the transition state (TS) structure, then you can find the respective transition state with the following keywords OptTS and ScanTS (in this example we take the structure with highest energy of the above relaxed surface scan):

```
! B3LYP SV(P) TightSCF SlowConv OptTS
# performs a TS optimization with the EF-algorithm
# Transition state: H-atom abstraction from CH4 to OH-radical

%geom
   Calc_Hess true  # calculation of the exact Hessian
                   #  before the first optimization step
end

* int 0 2
    C  0 0 0 0.000000   0.000   0.000
    H  1 0 0 1.285714   0.000   0.000
    H  1 2 0 1.100174 107.375   0.000
    H  1 2 3 1.100975 103.353 119.612
    H  1 2 3 1.100756 105.481 238.889
    O  2 1 3 1.244156 169.257  17.024
    H  6 2 1 0.980342 100.836  10.515
*
```

Note

- You need a good guess of the TS structure. Relaxed surface scans can help in almost all cases (see also example above).
- For TS optimization (in contrast to geometry optimization) an exact Hessian, a Hybrid Hessian or a modification of selected second derivatives is necessary.
- Analytic Hessian evaluation is available for HF and SCF methods, including the RI and RIJCOSX approximations and canonical MP2.
- Check the eigenmodes of the optimized structure for the eigenmode with a single imaginary frequency. You can also visualize this eigenmode with `orca_pltvib` (section *Animation of Vibrational Modes*) or any other visualization program that reads ORCA output files.

- If the Hessian is calculated during the TS optimization, it is stored as basename.001.hess, if it is recalculated several times, then the subsequently calculated Hessians are stored as basename.002.hess, basename.003.hess, …
- If you are using the Hybrid Hessian, then you have to check carefully at the beginning of the TS optimization (after the first three to five cycles) whether the algorithm is following the correct mode (see TIP below). If this is not the case you can use the same Hybrid Hessian again via the inhess read keyword and try to target a different mode (via the `TS_Mode` keyword, see below).

In the example above the TS mode is of local nature. In such a case you can directly combine the relaxed surface scan with the TS optimization with the

```
! ScanTS
```

command, as used in the following example:

```
! B3LYP SV(P) TightSCF SlowConv
! ScanTS
# perform a relaxed surface scan and TS optimization
#  in one calculation
%geom
     scan
        B 1 0 = 2.0, 1.0, 15
     end
end
* int 0 2
 C     0   0   0   0.000000     0.000     0.000
 H     1   0   0   1.999962     0.000     0.000
 H     1   2   0   1.095870   100.445     0.000
 H     1   2   3   1.095971    90.180   119.467
 H     1   2   3   1.095530    95.161   238.880
 O     2   1   3   0.984205   164.404    27.073
 H     6   2   1   0.972562   103.807    10.843
*
```

Note

- The algorithm performs the relaxed surface scan, aborts the Scan after the maximum is surmounted, chooses the optimized structure with highest energy, calculates the second derivative of the scanned coordinate and finally performs a TS optimization.
- If you do not want the scan to be aborted after the highest point has been reached but be carried out up to the last point, then you have to type:

```
%geom
  fullScan true # do not abort the scan with !ScanTS
end
```

As transition state finder we implemented the quasi-Newton like Hessian mode following algorithm.[531, 533, 534, 535, 536, 537, 538, 540, 541] This algorithm maximizes the energy with respect to one (usually the lowest) eigenmode and minimizes with respect to the remaining $3N - 7(6)$ eigenmodes of the Hessian.

Tip

- You can check at an early stage if the optimization will lead to the "correct" transition state. After the first optimization step you find the following output for the redundant internal coordinates:

```
-----------------------------------------------------------------------
                     Redundant Internal Coordinates
```

```
                          (Angstroem and degrees)
   Definition                        Value    dE/dq     Step    New-Value  comp.(TS␣
↪mode)
-----------------------------------------------------------------------------
 1. B(H   1,C   0)                   1.2857  0.013136  0.0286    1.3143          0.
↪58
 2. B(H   2,C   0)                   1.1002  0.014201 -0.0220    1.0782
 3. B(H   3,C   0)                   1.1010  0.014753 -0.0230    1.0779
 4. B(H   4,C   0)                   1.1008  0.014842 -0.0229    1.0779
 5. B(O   5,H   1)                   1.2442 -0.015421 -0.0488    1.1954          0.
↪80
 6. B(H   6,O   5)                   0.9803  0.025828 -0.0289    0.9514
 7. A(H   1,C   0,H   2)             107.38 -0.001418   -0.88    106.49
 8. A(H   1,C   0,H   4)             105.48 -0.002209   -0.46    105.02
 9. A(H   1,C   0,H   3)             103.35 -0.003406    0.08    103.43
10. A(H   2,C   0,H   4)             113.30  0.001833    0.35    113.65
11. A(H   3,C   0,H   4)             113.38  0.002116    0.26    113.64
12. A(H   2,C   0,H   3)             112.95  0.001923    0.45    113.40
13. A(C   0,H   1,O   5)             169.26 -0.002089    4.30    173.56
14. A(H   1,O   5,H   6)             100.84  0.003097   -1.41     99.43
15. D(O   5,H   1,C   0,H   2)        17.02  0.000135    0.24     17.26
16. D(O   5,H   1,C   0,H   4)      -104.09 -0.000100    0.52   -103.57
17. D(O   5,H   1,C   0,H   3)       136.64  0.000004    0.39    137.03
18. D(H   6,O   5,H   1,C   0)        10.52  0.000078   -0.72      9.79
-----------------------------------------------------------------------------
```

Every Hessian eigenmode can be represented by a linear combination of the redundant internal coordinates. In the last column of this list the internal coordinates, that represent a big part of the mode which is followed uphill, are labelled. The numbers reflect their magnitude in the TS eigenvector (fraction of this internal coordinate in the linear combination of the eigenvector of the TS mode). Thus at this point you can already check whether your TS optimization is following the right mode (which is the case in our example, since we are interested in the abstraction of H1 from C0 by O5.

- If you want the algorithm to follow a different mode than the one with lowest eigenvalue, you can either choose the number of the mode or you can give an internal coordinate that should be strongly involved in this mode using the `TS_Mode` keyword.

Tip

- If you look for a TS of a breaking bond the respective internal coordinate might not be included in the list of redundant internal coordinates due to the bond distance being slightly too large, leading to slow or even no convergence at all. In order to prevent that behavior a region of atoms that are active in the TS search can be defined, consisting of e.g. the two atoms of the breaking bond. During the automatic generation of the internal coordinates the bond radii of these atoms (and their neighbouring atoms) are increased, making it more probable that breaking or forming bonds in the TS are detected as bonds. See keywords `TS_Active_Atoms` and `TS_Active_Atoms_Factor` in the keywords list below.

Hessians for Transition State Calculations

For transition state (TS) optimization a simple initial Hessian, which is used for minimization, is not sufficient. In a TS optimization we are looking for a first order saddle point, and thus for a point on the PES where the curvature is negative in the direction of the TS mode (the TS mode is also called transition state vector, the only eigenvector of the Hessian at the TS geometry with a negative eigenvalue). Starting from an initial guess structure the algorithm used in the ORCA TS optimization has to climb uphill with respect to the TS mode, which means that the starting structure has to be near the TS and the initial Hessian has to account for the negative curvature of the PES at that point. The simple force-field Hessians cannot account for this, since they only know harmonic potentials and thus positive curvature.

The most straightforward option in this case would be (after having looked for a promising initial guess structure with the help of a relaxed surface scan) to calculate the exact Hessian before starting the TS optimization. With this Hessian (depending on the quality of the initial guess structure) we know the TS eigenvector with its negative eigenvalue and we have also calculated the exact force constants for all other eigenmodes (which should have positive force constants). For the HF, DFT methods and MP2, the analytic Hessian evaluation is available and is the best choice, for details see section Frequencies (*Vibrational Frequencies*).

When only the gradients are available (most notably the CASSCF), the numerical calculation of the exact Hessian is very time consuming, and one could ask if it is really necessary to calculate the full exact Hessian since the only special thing (compared to the simple force-field Hessians) that we need is the TS mode with a negative eigenvalue.

Here ORCA provides two different possibilities to speed up the Hessian calculation, depending on the nature of the TS mode: the Hybrid Hessian and the calculation of the Hessian value of an internal coordinate. For both possibilities the initial Hessian is based on a force-field Hessian and only parts of it are calculated exactly. If the TS mode is of very local nature, which would be the case when e.g. cleaving or forming a bond, then the exactly calculated part of the Hessian can be the second derivative of only one internal coordinate, the one which is supposed to make up the TS mode (the formed or cleaved bond). If the TS mode is more complicated and more delocalized, as e.g. in a concerted proton transfer reaction, then the hybrid Hessian, a Hessian matrix in which the numerical second derivatives are calculated only for those atoms, which are involved in the TS mode (for more details, see section *Geometry Optimizations*), should be sufficient. If you are dealing with more complicated cases where these two approaches do not succeed, then you still have the possibility to start the TS optimization with a full exact Hessian.

Numerical Frequency calculations are quite expensive. You can first calculate the Hessian at a lower level of theory or with a smaller basis set and use this Hessian as input for a subsequent TS optimization:

```
%geom
  inhess   Read                    # this command comes with:
  InHessName "yourHessian.hess"    # filename of Hessian input file
end
```

Another possibility to save computational time is to calculate exact Hessian values only for those atoms which are crucial for the TS optimization and to use approximate Hessian values for the rest. This option is very useful for big systems, where only a small part of the molecule changes its geometry during the transition and hence the information of the full exact Hessian is not necessary. With this option the coupling of the selected atoms are calculated exactly and the remaining Hessian matrix is filled up with a model initial Hessian:

```
%geom
  Calc_Hess true
  Hybrid_Hess {0 1 5 6} end   # calculates a Hybrid Hessian with
                              # exact calculation for atoms 0, 1, 5 and 6
end
```

For some molecules the PES near the TS can be very far from ideal for a Newton-Raphson step. In such a case ORCA can recalculate the Hessian after a number of steps:

```
%geom
  Recalc_Hess  5   # calculate the Hessian at the beginning
                   # and recalculate it after 5,10,15,... steps
end
```

Another solution in that case is to switch on the trust radius update, which reduces the step size if the Newton-Raphson steps behave unexpected and ensures bigger step size if the PES seems to be quite quadratic:

```
%geom
  Trust 0.3 # Trust <0 - use fixed trust radius (default: -0.3 au)
            # Trust >0 - use trust radius update, i.e. 0.3 means:
            # start with trust radius 0.3 and use trust radius update
end
```

Special Coordinates for Transition State Optimizations

- If you look for a TS of a breaking bond the respective internal coordinate might not be included in the list of redundant internal coordinates (but this would accelerate the convergence). In such a case (and of course in others) you can add coordinates to or remove them from the set of autogenerated redundant internal coordinates (alternatively check the TS_Active_Atoms keyword):

```
# add ( A ) or remove ( R ) internal coordinates
%geom
  modify_internal
      { B 10 0 A }      # add a bond between atoms 0 and 10
      { A 8 9 10 R }    # remove the angle defined
                        # by atoms 8, 9 and 10
      { D 7 8 9 10 R }  # remove the dihedral angle defined
  end                   # by atoms 7, 8, 9 and 10
end
```

4.3.3 ScanTS option

For TS modes of rather local nature (involving only one bond or an angle; no concerted movements over multiple atoms) we implemented the ScanTS feature. Here the user can carry out a relaxed surface scan and a TS optimization in one calculation. After the relaxed surface scan the algorithm chooses the optimized structure of the scan with highest energy as initial guess structure and the two neighbouring structures for the calculation of the second derivative of the scanned coordinate (e.g., if scan step number 4 gives the structure with highest energy, then structure `basename.004.xyz` is the initial guess for the TS optimization; the structures `basename.003.xyz` and `basename.005.xyz` are used for the calculation of the second derivative). Before the first step of the subsequent TS optimization the energies and gradients for all three structures are calculated. The gradients are then transformed to internal coordinates. The diagonal Hessian value of the scanned coordinate is then calculated via finite difference of the internal gradients of the two given structures (003 and 005 in our example).

For the construction of the initial Hessian a model force field Hessian is built up (this Hessian has got only diagonal entries and zeros as off-diagonal elements). The exactly calculated diagonal Hessian value replaces the model force field Hessian entry for the respective internal coordinate.

If the user already performed a regular relaxed surface scan without the subsequent TS optimization, then he can nevertheless use these structures for the same procedure. A relaxed surface scan always gives you the xyz-files and gbw-files of the optimized structures of each scan step. A separate TS optimization can be carried out where the structure with highest energy is the starting structure. Additionally the two files with the two adjacent structures (as explained above) have to be provided (via the Hess_Internal keyword, see below). Furthermore, the internal coordinate, for which the diagonal Hessian value has to be calculated, has to be given (the previously scanned coordinate). This exact Hessian calculation is only possible for one internal coordinate:

```
%geom
  Hess_Internal
     {B 1 0 C}                # previously scanned coordinate
     XYZ1 "scanName.003.xyz" # the xyz-files of the structures
     XYZ2 "ScanName.005.xyz" #  next to the highest energy point
     GBW1 "ScanName.003.gbw" # the gbw-files of the structures
     GBW2 "ScanName.005.xyz" #  next to the highest energy
```

(continues on next page)

(continued from previous page)

```
                                # the gbw-files are optional
   end
end
```

Additionally the manipulation of the diagonal Hessian values of the internal Hessian is possible for further internal coordinates, but without an extra calculation. Here the user can just define a value (in Eh/Bohr^2).

```
Hess_Internal
  {A 3 2 1 D 2.0} # define a diagonal Hessian value of
                  #  2 Eh/Bohr2 for the angle between
                  #  atoms 3 2 1
  {B 1 0 D -0.5}  # define a diagonal Hessian value of
                  #  -0.5 Eh/Bohr2 for the bond between
                  #  atoms 1 and 0
end
```

The definition of such Hessian (diagonal) elements is possible for multiple internal coordinates. These just replace the values of the force field model Hessian.

4.3.4 Hybrid Hessian

We implemented the calculation of a "Hybrid Hessian" as an alternative to the full Hessian calculation for TS optimization. Here only those parts of the Hessian, that are important for the TS optimization, are calculated exactly. For this calculation we define two kinds of atoms: atoms whose couplings with the other atoms are treated exactly (E) and atoms whose couplings are treated approximately (A).

In a first step an Almloef model Hessian is built up in redundant internal coordinates and transformed to Cartesian coordinates. This Hessian gives the second derivative elements for atom pairs A/A. In a second step the second derivative elements between pairs E/E and E/A are calculated numerically as in a numerical frequency calculation:

$$\frac{\partial^2 \mathbf{E}}{\Delta i_B \Delta j_C} = \frac{\partial^2 \mathbf{E}}{\Delta j_C \Delta i_B} = \frac{g_{j,C}^{i,B} - g_{j,C}^{eq.}}{displ.} \tag{4.21}$$

with:

i, j	x-, y- or z-direction
B, C	pairs of E/E, E/A, A/E
$displ.$	magnitude of displacement
$g_{j,C}^{eq.}$	force on atom C in direction j in current geometry
$g_{j,C}^{i,B}$	force on atom C in direction j after displacement of atom B in direction i

4.3.5 Partial Hessian Vibrational Analysis

We implemented the Partial Hessian Vibrational Analysis (PHVA), as published by Li, Jensen in [531], for the analysis of the nature of stationary points of structures obtained with QM/MM optimizations.

```
# PHVA after a QM/MM optimization in the (dispersion-/PC-) field
# caused by the MM-atoms
! NumFreq
%LJCoefficients "temp.LJ" # file with the Lennard Jones
                          #  coefficients for dispersion interaction
                          # obtained from last QM/MM run
%pointcharges "temp.pc"   # file with the point charges for
```

(continues on next page)

(continued from previous page)

```
                          #  electrostatic interaction
                          # obtained from last QM/MM run
 #
%freq
    PARTIAL_Hess {0 1 2}  # atoms which are "frozen" and which make
                          #  the boundary to the MM-system
     end
   end
```

Note

- This procedure should be used for QM/MM optimized structures only to verify the nature of the stationary point and have an estimate of the ZPE.

Here we shortly describe the procedure: In PHVA we divide the system into two parts B (of size n atoms) and C (size $N-n$). Let the atom set B belong to the region where the chemical change is localized. The Partial Hessian matrix is built up as follows:

$$\begin{pmatrix} K_{BB} & 0 \\ 0 & K_{CC}^{\varepsilon} \end{pmatrix} \tag{4.22}$$

With:

$$K_{BB} \quad : x, y, z \quad \text{direction}$$

$$K_{CC}^{\varepsilon} = \begin{pmatrix} \varepsilon & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}, \varepsilon = 10^{-8}\,\text{au}, \tag{4.23}$$

this corresponds to using near-infinite masses for the atoms in C.

With this procedure we get the following eigenvalue structure:

- Six zero eigenvalues with modes corresponding to translational and rotational motion of the entire molecule.
- $3(N-n)-6$ small (less than 1 cm^{-1}) eigenvalues with modes corresponding mainly to internal motion within region C.
- Three eigenvalues (typically less than 10 cm^{-1}) with modes corresponding mainly to motion of region C relative to region B.
- $(3n-3)$ eigenvalues with modes corresponding mainly to relative motion of B and C as well as internal motion within region B.

4.3.6 Tranistion State Theory Keywords

Table 4.5: `%geom` block input keywords and options for transition state geometry optim

Keyword	Option / Value	Description
`OptTS`		Transition structure optimization. Included as `!OptTS`
`Scan`		Must include `!ScanTS` for transition state scan.
	`B 1 2 = v1, v2, N`	Scan atom 1 to atom 2 bond distance from v1 to v2 in N st
	`B 1 2 [v1 v2 ...]`	Scan atom 1 to atom 2 bond with custom step values.
	`A 1 2 3`	Scan an angle using atoms 1,2,3.
	`D 1 2 3 4`	Scan a dihedral of atoms 1,2,3,4.
`Simul_Scan`	`true`	Enable simultaneous multidimensional scans (up to 3 coord
`FullScan`	`true`	Forces full surface scan before TS optimization (`!ScanTS`

Table 4.5 – continued from previous page

Keyword	Option / Value	Description
TS_search	EF	Enable TS search using Eigenvector Following.
TS_Mode	{M 0}	Follow eigenvector 0 (lowest mode).
	{B 0 1}	Follow bond between atoms 0 and 1.
	{A 2 1 0}	Follow angle between atoms 2, 1, and 0.
	{D 3 2 1 0}	Follow dihedral between atoms 3, 2, 1, and 0.
TS_Active_Atoms	{ 1 2 3 }	Atoms that are involved in TS, e.g. for proton transfer the
TS_Active_Atoms_Factor	1.5	Factor by which the cutoff for bonds is increased for the ab
inhess	unit	Start optimization with a unit matrix Hessian.
	Read	Read Hessian from external file. Requires InHessName.
InHessName	"filename.hess"	Specifies file containing the Hessian (e.g., .hess, .opt,
	Lindh	Use Lindh's model Hessian (only for redundant coordinate
	Almloef	Use Almloef's model Hessian.
	Schlegel	Use Schlegel's model Hessian.
	Swart	Use Swart and Bickelhaupt's model Hessian.
	XTB0	Use GFN0-xTB Hessian.
	XTB1	Use GFN1-xTB Hessian.
	XTB2	Use GFN2-xTB Hessian.
	GFNFF	Use GFN-FF Hessian.
Calc_Hess	true	Calculate numerical Hessian at the beginning of optimizati
Recalc_Hess	5	Recalculate Hessian every 5 cycles.
Hybrid_Hess	{0 1 5 6} end	Calculate exact Hessian for specified atoms, use model for
NumHess	true	Request numerical Hessian.
Hess_Internal	{A 3 2 1 D 2.0}	Manually set Hessian value (Eh/Bohr^2) for angle between a
	reset 5	Reset modified Hessian values after 5 cycles.
	{B 1 0 C}	Coordinates for Hessian refinement after a relaxed surface
	XYZ1 "scanName.003.xyz"	Structure file before maximum in scan.
	XYZ2 "ScanName.005.xyz"	Structure file after maximum in scan.
	GBW1 "ScanName.003.gbw"	Optional GBW file before max in scan.
	GBW2 "ScanName.005.xyz"	Optional GBW file after max in scan.
Update	Powell	Use Powell update for Hessian.
	Bofill	Use Bofill update (default for TS optimization).
	BFGS	Use BFGS update (default for geometry optimization).
HESS_Modification	Shift_Diag	Shift diagonal elements of Hessian (default).
	EV_Reverse	Reverse sign of diagonal elements.
HESS_MinEV	0.0001	Minimum allowed Hessian eigenvalue.
NResetHess	20	Rebuild model Hessian after 20 steps (BFGS only).
NStepsInResetHess	5	Number of previous steps used to inform new model Hessi
Trust	-0.3	Use fixed trust radius of 0.3 au (negative sign indicates fixe
modify_internal	{B 10 0 A}	Add bond between atoms 0 and 10.
	{A 8 9 10 R}	Remove angle between atoms 8, 9, and 10.
	{D 7 8 9 10 R}	Remove dihedral between atoms 7, 8, 9, and 10.

4.4 Intrinsic Reaction Coordinate

The Intrinsic Reaction Coordinate (IRC) is a special form of a minimum energy path, connecting a transition state (TS) with its downhill-nearest intermediates. A method determining the IRC is thus useful to determine whether a transition state is directly connected to a given reactant and/or a product. The implementation in ORCA follows the method suggested by Morokuma and coworkers.[542]

The IRC method follows the gradient of the nuclear coordinates. As the gradient is negligible at a TS, first an initial displacement from the TS structure has to be carried out, based on the eigenmodes of the Hessian, in order to get to a region with a non-negligible gradient. For the initial displacement the eigenvector of the eigenmode with the lowest frequency (hessMode=0) is normalized and then scaled by Scale_Displ_SD (which by default is chosen such that an

energy change of Init_Displ_DE can be expected). Two initial displacements, forward and backward, are taken by adding the resulting displacement vector (multiplied with +1 and -1, respectively) to the initial structure. If the user requests the downhill direction (e.g. from a previous unconverged IRC run), it is assumed that the gradient is nonzero and thus no initial displacement is carried out.

After the initial displacement, the iterations of the IRC method begin. Each iteration consists of two main steps, each of which consists again of multiple SP and gradient runs:

1. Initial steepest descent (SD) step:
    1. The gradient (grad0) of the starting geometry (G0) is normalized, scaled by Scale_Displ_SD, and the resulting displacement vector (SD1) is applied to G0.
    2. Optional (if SD_ParabolicFit is true): If SD1 increases the energy, a linear search is taken along the direction of the displacement vector:
        1. The displacement vector SD1 is scaled by 0.5 (SD2 = 0.5 x SD1) and again added to G0.
        2. A parabolic fit for finding the displacement vector (SD3) which leads to minimal energy is carried out using the three SP energies (G0, geometry after SD1 and after SD2 step). SD3 has the same direction as SD1 and SD2, but can have a different length.
        3. The keyword Interpolate_only controls whether the length of SD3 has to be in between 0 and the length of SD1. If that is the case, the maximum length is determined by SD1, the minimum length is zero.
    3. At the resulting geometry G1 (G0+SD1 or G0+SD3) the gradient is calculated (grad1).
2. Optional (if Do_SD_Corr is true): Correction to the steepest descent step:
    1. Based on grad0 and grad1 a vector is computed which represents a correction to the first SD (SD1 or SD3) step. This correction brings the geometry closer to the IRC.
    2. This vector is normalized, scaled by Scale_Displ_SD_Corr times the length of SD1 or SD3, and the resulting displacement vector (SDC1) is applied to G1.
    3. Optional (if SD_Corr_ParabolicFit is true):
        1. If the energy increases after applying step SDC1, SDC1 is scaled by 0.5 (SDC2 = 0.5 x SD1), if the energy decreases, SDC1 is scaled by 2 (SDC2 = 2 x SD1). SDC2 is then added to G1.
        2. A parabolic fit for finding the displacement vector (SDC3) which leads to minimal energy is carried out using the three SP energies (G1, geometry after SDC1 and after SDC2 step). SDC3 has the same direction as SDC1 and SDC2, but can have a different length.
        3. The keyword Interpolate_only controls whether the length of SDC3 has to be in between 0 and the length of SDC1. If that is the case, the maximum length is determined by SDC1, the minimum length is zero.
    4. At the resulting geometry G2 (G1+SDC1 or G1+SDC3) the gradient is calculated (grad2).
3. The gradient at the new geometry is checked for convergence.
4. Optional (if Adapt_Scale_Displ is true):
    1. If the resulting overall step size is smaller than 0.5 times Scale_Displ_SD, Scale_Displ_SD is multiplied by 0.5.
    2. If the resulting overall step size is larger than 2 times Scale_Displ_SD, Scale_Displ_SD is multiplied by 2.
    3. Scale_Displ_SD may not become smaller than 1/16 times the initial Scale_Displ_SD and not larger than 4 times the initial Scale_Displ_SD.

4.4.1 Basic Usage

The following keywords are available:

```
! B3LYP SV(P) TightSCF KDIIS SOSCF Freq IRC
* xyz 0 2
  C  -0.000   0.001   -0.000
  H   1.290   0.005   -0.006
  H  -0.330   1.050   -0.002
  H  -0.252  -0.532   -0.929
  H  -0.286  -0.545    0.911
  O   2.499   0.220    0.065
  H   2.509   1.085    0.525
*
```

Note

- The same method and basis set as used for optimization and frequency calculation should be used for the IRC run.
- The IRC keyword can be requested without, but also together with OptTS, ScanTS, NEB-TS, AnFreq and NumFreq keywords.
- In its default settings the IRC code checks whether a Hessian was computed before the IRC run. If that is not the case, and if no Hessian is defined via the %irc block, a new Hessian is computed at the beginning of the IRC run.
- A final trajectory (_IRC_Full_trj.xyz) is generated which contains both directions, forward and backward, by starting from one endpoint and going to the other endpoint, visualizing the entire IRC. Forward (_IRC_F_trj.xyz and _IRC_F.xyz) and backward (_IRC_B_trj.xyz and _IRC_B.xyz) trajectories and xyz files contain the IRC and the last geometry of that respective run.

4.4.2 Keywords

```
! IRC
%irc
   MaxIter    20
   PrintLevel 1
   Direction  both # both - default
                   # forward
                   # backward
                   # down
 # Initial displacement
   InitHess   read # by default ORCA uses the Hessian from AnFreq or NumFreq, or
                   # computes a new one
                   # read    - reads the Hessian that is defined via Hess_Filename
                   # calc_anfreq  - computes the analytic Hessian
                   # calc_numfreq - computes the numeric Hessian
   Hess_Filename "h2o.hess"  # input Hessian for initial displacement, must be used
                             # together with InitHess = read
   hessMode   0  # Hessian mode that is used for the initial displacement. Default␣
↪0
   Init_Displ_DE      # DE (default) - energy difference
                      # length       - step size
   Scale_Init_Displ 0.1 # step size for initial displacement from TS. Default 0.1␣
↪a.u.
   DE_Init_Displ    2.0 # energy difference that is expected for initial␣
↪displacement
                        #  based on provided Hessian (Default: 2 mEh)
```

(continues on next page)

(continued from previous page)

```
 # Steps
   Follow_CoordType cartesian # default and only option
   Scale_Displ_SD       0.15  # Scaling factor for scaling the 1st SD step
   Adapt_Scale_Displ    true  # modify Scale_Displ_SD when the step size becomes
                              # smaller or larger
   SD_ParabolicFit      true  # Do a parabolic fit for finding an optimal SD step
                              # length
   Interpolate_only     true  # Only allow interpolation for parabolic fit, not
                              # extrapolation
   Do_SD_Corr           true  # Apply a correction to the 1st SD step
   Scale_Displ_SD_Corr  0.333 # Scaling factor for scaling the correction step to
                              # the SD step. It is multiplied by the length of the
                              # final 1st SD step
   SD_Corr_ParabolicFit true  # Do a parabolic fit for finding an optimal␣
↪correction
                              # step length
 # Convergence thresholds - similar to LooseOpt
   TolRMSG   5.e-4      # RMS gradient (a.u.)
   TolMaxG   2.e-3      # Max. element of gradient (a.u.)
 # Output options
   Monitor_Internals # Up to three internal coordinates can be defined
      {B 0 1}        # for which the values are printed during the IRC run.
      {B 1 5}        # Possible are (B)onds, (A)ngles, (D)ihedrals and (I)mpropers
   end
 end
```

NOTE

- For direction=down (downhill) no initial displacement is necessary, and thus no Hessian is needed.

4.5 Nudged Elastic Band Method

The Nudged Elastic Band (NEB) method is used to find a minimum energy path (MEP) connecting given reactant and product state minima on the energy surface. An initial path is generated and represented by a discrete set of configurations of the atoms, referred to as images of the system. The number of images is specified by the user and has to be large enough to obtain sufficient resolution of the path. The implementation in ORCA is described in detail in the article by Ásgeirsson et. al.[543] and in section *Nudged Elastic Band Method* along with the input options. The most common use of the NEB method is to find the highest energy saddle point on the potential energy surface specifying the transition state for a given initial and final state. Rigorous convergence to a first-order saddle point can be obtained with the climbing image NEB (CI-NEB), where the highest energy image is pushed uphill in energy along the tangent to the path while relaxing downhill in orthogonal directions. Another method for finding a first-order saddle point is the NEB-TS which uses the CI-NEB method with a loose tolerance to begin with and then switches over to the OptTS method to converge on the saddle point. This combination can be a good choice for calculations of complex reactions where the ScanTS method fails or where 2D relaxed surface scans are necessary to find a good initial guess structure for the OptTS method. The zoomNEB variants are a good choice in case of very complex transition states with long tails. For more detailed information on the various NEB variants implemented in ORCA please consult section *Nudged Elastic Band Method*.

In their simplest form NEB, NEB-CI and NEB-TS only require the reactant and product state configurations (one via the xyz block, and the other one via the keyword neb_end_xyzfile):

```
!NEB-TS # or !NEB or !NEB-CI or !ZOOM-NEB-TS or !ZOOM-NEB-CI
        # or !Fast-NEB-TS  (corresponds to IDPP-TS defined in the NEB-TS␣
↪manuscript)
        # or !Loose-NEB-TS (corresponds to default NEB-TS in the NEB-TS manuscript)
%neb
 neb_end_xyzfile "final.xyz"
end
```

Below is an example of an NEB-TS run involving an intramolecular proton transfer within acetic acid. The simplest input is

```
!XTB NEB-TS
%neb
 neb_end_xyzfile "final.xyz"
end

*xyz 0 1
C       0.416168     0.038758    -0.014077
C       0.041816     0.011798     1.439610
O       1.524458     0.176600    -0.453888
O      -0.654209    -0.127881    -0.803857
H      -0.391037    -0.126036    -1.737478
H      -0.913438     0.507022     1.585301
H      -0.057787    -1.026455     1.750845
H       0.819515     0.485425     2.030252
*
```

Where the final.xyz structure contains the corresponding structure with the proton on the other oxygen.

The initial path is reasonable and the CI calculation can be switched on after five NEB iterations.

```
Starting iterations:

Optim.  Iteration  HEI  E(HEI)-E(0)  max(|Fp|)   RMS(Fp)    dS
Switch-on CI threshold               0.020000
   LBFGS     0      4    0.081017    0.073897   0.018915  3.2882
   LBFGS     1      5    0.070244    0.056668   0.013913  3.2770
   LBFGS     2      5    0.062934    0.038972   0.008763  3.3376
   LBFGS     3      5    0.057358    0.032076   0.006535  3.3950
   LBFGS     4      4    0.053260    0.019015   0.003599  3.4826

Image  4 will be converted to a climbing image in the next iteration (max(|Fp|) <␣
↪0.0200)

Optim.  Iteration  CI   E(CI)-E(0)   max(|Fp|)   RMS(Fp)    dS     max(|FCI|)  ␣
↪RMS(FCI)
Convergence thresholds               0.020000   0.010000            0.002000    0.
↪001000
```

The CI run converges after another couple of iterations:

```
*********************H U R R A Y*********************
***         THE NEB OPTIMIZATION HAS CONVERGED      ***
*****************************************************
```

Subsequently a summary of the MEP is printed:

```
-----------------------------------------------------------
                         PATH SUMMARY
-----------------------------------------------------------
All forces in Eh/Bohr.

Image Dist.(Ang.)    E(Eh)    dE(kcal/mol)  max(|Fp|)  RMS(Fp)
  0     0.000     -14.45993      0.00       0.00011   0.00004
  1     0.426     -14.44891      6.91       0.00092   0.00033
  2     0.652     -14.42864     19.63       0.00084   0.00038
  3     0.805     -14.41132     30.50       0.00075   0.00027
  4     0.932     -14.40562     34.08       0.00057   0.00018 <= CI
  5     1.044     -14.41047     31.03       0.00057   0.00024
  6     1.153     -14.42200     23.80       0.00103   0.00034
  7     1.280     -14.43666     14.60       0.00098   0.00037
```

(continues on next page)

(continued from previous page)

```
  8     1.476     -14.45106      5.56       0.00106   0.00033
  9     1.869     -14.45988      0.03       0.00013   0.00006
```

Additionally, detailed information on the highest energy image (or the CI) is printed:

```
-----------------------------------------------------------------
                INFORMATION ABOUT SADDLE POINT
-----------------------------------------------------------------

Climbing image                              ....  4
Energy                                      ....  -14.40561577 Eh
Max. abs. force                             ....  9.5976e-04 Eh/Bohr

----------------------------------------
  SADDLE POINT (ANGSTROEM)
----------------------------------------
C      0.040867     0.007347    -0.497635
C     -0.075595     0.017879     0.979075
O      1.122340     0.126074    -1.145534
O     -0.928470    -0.137946    -1.298318
H      0.165808    -0.021676    -2.055704
H     -0.996979     0.514720     1.271668
H     -0.116377    -1.013504     1.327873
H      0.788406     0.507105     1.418575

----------------------------------------
  FORCES (Eh/Bohr)
----------------------------------------
C     -0.000646    -0.000111     0.000086
...

----------------------------------------
  UNIT TANGENT
----------------------------------------
C     -0.246569    -0.031821    -0.019359
...

 => Unit tangent is an approximation to the TS mode at the saddle point

Next a TS optimization is performed on the CI from the NEB run.
```

Finally, a TS optimization is started, after which the MEP information is updated by including the TS structure:

```
-----------------------------------------------------------------
                      PATH SUMMARY FOR NEB-TS
-----------------------------------------------------------------
All forces in Eh/Bohr. Global forces for TS.

Image      E(Eh)    dE(kcal/mol)  max(|Fp|)  RMS(Fp)
  0     -14.45993      0.00       0.00011   0.00004
  1     -14.44891      6.91       0.00092   0.00033
  2     -14.42864     19.63       0.00084   0.00038
  3     -14.41132     30.50       0.00075   0.00027
  4     -14.40562     34.08       0.00057   0.00018 <= CI
 TS     -14.40562     34.08       0.00033   0.00013 <= TS
  5     -14.41047     31.03       0.00057   0.00024
  6     -14.42200     23.80       0.00103   0.00034
  7     -14.43666     14.60       0.00098   0.00037
  8     -14.45106      5.56       0.00106   0.00033
  9     -14.45988      0.03       0.00013   0.00006
```

Note that here both TS and CI are printed for comparison.

4.6 Nudged Elastic Band Method

The Nudged Elastic Band (NEB)[544, 545, 546] method is used to find a minimum energy path (MEP) connecting two local energy minima on the potential energy surface (PES) and thereby an estimate of the activation energy for the transition. The two minima are referred to as the reactant and product in the following discussion. The path can have one or more maxima, each one corresponding to a first order saddle point on the energy surface. The NEB method offers an advantage over eigenvector-following methods in that it is guaranteed to find saddle points that connect the given reactant and product states. The minimum energy path is often used to represent the reaction coordinate of the transition between the two states. The methods and implementation outlined below are discussed further in Ref. [543].

The user needs to specify the reactant and product configurations. The reactant energy minimum is inserted into the regular ORCA input file while the product should be in a separate .xyz file (keyword: *neb_end_xyzfile*). The reactant and product configurations should be optimized *a priori* by relaxing to energy minima on the PES, see section *Geometry Optimizations*. This can also be achieved via the 'preopt_ends' keyword. It is important to carefully prepare the reactant and product such that the position (or index) of the atoms is the same in the two configuration files, i.e. there should be one-to-one mapping between the reactant and product configurations.

The discretized path between the two minima is represented by a set of M system configurations that are referred to as images, i.e. $\mathbf{R} = [r_0, r_1, ..., r_{M-1}]$. The number of intermediate images between the end points (i.e. reactant and product) is specified by the user. The general rule of thumb is to include $M = 7 - 12$, or $5 - 10$ intermediate images per energy maximum in order to obtain a high enough resolution of the path and the saddle point. However, calculations can often converge and give accurate results with fewer images but complex paths with multiple maxima or long tails may require more images. During an NEB calculation the intermediate images are iteratively shifted towards the MEP using the component of the atomic force that is perpendicular to the current path as estimated from the tangent to the path at each image, while the endpoint images are typically kept fixed. In each step of the iterative process, the energy and atomic forces of each intermediate image need to be computed. One of the main advantages of the NEB method is that the calculations of the images are carried out in parallel, where the electronic structure computations can be distributed over multiple processors (see discussion below for more details on the parallelization). While the CPU time is proportional to the number of images, the number of iterations needed for convergence to the MEP can become smaller when more images are included in the discrete representation of the path.

The tangent to the path at each image, $\hat{\tau}_i$, can be estimated in two ways (keyword: *tangent*), either by the original method [545] or by the more numerically stable improved [546] estimate (default option). In the former, the tangent at an image is taken to be a linear combination of the two line segments that connect to image i,

$$\tau_i = \frac{r_{i+1} - r_i}{|r_{i+1} - r_i|} + \frac{r_i - r_{i-1}}{|r_i - r_{i-1}|}$$

In the improved tangent estimate, the line segment from the current image to the adjacent image with higher energy is used, i.e.

$$\tau_i = \begin{cases} \tau^+ & \text{if } \mathrm{E}_{i+1} > \mathrm{E}_i > \mathrm{E}_{i-1} \\ \tau^- & \text{if } \mathrm{E}_{i+1} < \mathrm{E}_i < \mathrm{E}_{i-1} \end{cases}$$

where $\tau^+ = r_{i+1} - r_i$ and $\tau^- = r_i - r_{i-1}$. With the exception of image i being at an energy extremum along the path, then an energy-weighted average of the two line segments to adjacent images is used,

$$\tau_i = \begin{cases} \tau^+ \Delta V_i^{\max} + \tau^- \Delta V_i^{\min} & \text{if } \mathrm{E}_{i+1} > \mathrm{E}_{i-1} \\ \tau^+ \Delta V_i^{\min} + \tau^- \Delta V_i^{\max} & \text{if } \mathrm{E}_{i+1} < \mathrm{E}_{i-1} \end{cases}$$

where

$$\Delta V_i^{\max} = \max(|E_{i+1} - E_i|, |E_{i-1} - E_i|)$$
$$\Delta V_i^{\max} = \min(|E_{i+1} - E_i|, |E_{i-1} - E_i|)$$

Then the tangent is normalized, $\hat{\tau}_i = \tau_i / |\tau_i|$. With an accurate estimate of the tangent, the perpendicular component of the atom force is computed by,

$$F_i^{\perp} = F_i - (F_i \cdot \hat{\tau}_i)\, \hat{\tau}_i$$

In the free-end NEB method, the endpoint images are optimized simultaneously along with the intermediate images, i.e. M movable images are used. Three different variants of the free-end NEB method (keywords:*free_end*, *free_end_type*) are included in ORCA: where (i) the endpoints are restrained to move along the same (or separate) energy isocontour[547, 548], (ii) according to the atomic force acting perpendicular to the path and (iii) the full atomic force. For the first variant, the 'reactant' and 'product' endpont images, can be restrained to move along two different energy contours, E_0 (given by keyword:*free_end_ec*) and E_1 (given by keyword: *free_end_ec_end*), to keep the path bounded. Then, if the images drift away from the isocontour because of curvature of the energy surface, an harmonic restraint term is used to pull the image back to the contour.[548] The stiffness of the harmonic restraint is given by a user-supplied parameter (keyword:*free_end_kappa*). It is important to carefully select the energy values and the strength of the harmonic restraint, otherwise the path may become kinked. For the second variant, the endpoint images are unbounded but displaced directly towards the MEP. This is typically acceptable if the endpoint images are already in vicinity to the MEP and is less prone to kinks developing along the path. For the third variant, the endpoint images become optimized to the reactant and product energy minimum.

4.6.1 Spring forces

In order to control the distribution of the images along the path, spring forces are included to act between adjacent images, tangential to the path[545],

$$F_i^{\text{sp},\|} = \left(k_i^{\text{sp}}|r_{i+1} - r_i| - k_{i-1}^{\text{sp}}|r_i - r_{i-1}|\right)\hat{\tau}_i$$

The magnitude of the spring forces (i.e. stiffness) is controlled by spring constants, k^{sp}. The typical values of the spring constant (keyword: *springconst*) can be taken to be within the range of 0.01 Eh/Bohr^2 to 1.0 Eh/Bohr^2. If the spring constants are choosen to be the same for all pairs of adjacent images, the images will be equally distributed along the path. However, it is also possible to choose energy-weighted spring constants (keyword: *energy_weighted*) so as to increase the density of images in the higher energy regions[543, 549]. In an energy-weighted NEB calculation the spring constants are scaled according to the relative energy of the images, from a lower-bound value (keyword:*springconst*) to an upper-bound value (keyword:*springconst2*), by

$$k_i^{\text{sp}} = \begin{cases} (1-\alpha_i)k_{\text{u}} + \alpha_i k_{\text{l}}, & \text{if} E_i > E_{\text{ref}} \\ k_l, & \text{otherwise} \end{cases}$$
$$\alpha_i = \frac{E_{\text{max}} - E_i}{E_{\text{max}} - E_{\text{ref}}}$$

where k_{u} and k_{l} are the upper- and lower-bound values for the spring constant. E_{max} is the current estimate of the maximum energy along the path, E_i is the higher energy image of the pair of images connected by line segment i and E_{ref} is the energy of the higher energy minimum. The energy-weighted springs will typically serve to improve the tangent estimate in the barrier region and hence stabilize the calculations. The inclusion of energy-weighted springs can be important in reactions where the energy barrier is narrow and/or the pathway is characterized by a long 'energy tail', e.g., in rearrangements or dissociation reactions. The choice of spring constants will affect the behavior of a calculation, especially the number of iterations needed to reach convergence. Other formulations for spring forces are also available since ORCA 4.2 (keyword: *springtype*). These are referred to as the original[545] and ideal[550] spring forces. The original spring forces are estimated by a spring acting on each degree of freedom between a pair of adjacent images. For ideal springs each image is assigned an ideal position along the path based on a linear interpolation of the current location of the images and the individual images interact with the ideal locations via spring forces. The ideal springs are currently not implemented to work with energy weighted spring constants.

While only the component of the spring force parallel to the path is included in an NEB calculation (by default) the user can choose to include a fraction of the spring force acting perpendicular to the path to stabilize calculations (keyword: *perpspring*. Since, the perpendicular component of the spring force can serve to straighten out the path and is useful for complex pathways with multiple energy extrema and low resolution of the path. The inclusion of the perpendicular component of the spring force is always accompanied by a switching function that is used to scale it according to (i) convergence to the MEP (the 'tan' function)[551] (ii) the angle between adjacent images (the 'cos' function)[545] or a combination of both. The inclusion of the perpendicular spring force can help to reduce the number of iterations and eliminate kinks on the path. However, it is important to stress that by inclusion of the perpendicular spring forces, the images may not converge on the MEP. Alternatively, the user can choose to use the modified DNEB method[551, 552].

4.6.2 Optimization and convergence of the NEB method

The effective force used in a standard NEB calculations is the sum of the atomic force component perpendicular to the path and the spring force component parallel to the path,

$$F_i^{\text{NEB}} = F_i^{\perp} + F_i^{\text{sp},\parallel}$$

The path can be brought to the MEP by moving according to the effective force, as is shown in Fig. 4.2.

Fig. 4.2: Visualization of the effective force, F^{NEB} and its two components: $F_i^{\perp}$ and $F_i^{\text{sp},\parallel}$ for three images along an intermediate path in a NEB optimization. The figure is taken from Ref. [553]

This can be achieved by using any of the three optimization methods implemented (keyword:*opt_method*): velocity projection optimization (VPO)[545], fast inertial relxation engine (FIRE)[554] and L-BFGS[555]. VPO and FIRE are more robust for regions that are far from the MEP, while L-BFGS converges faster when the images are close to the MEP. FIRE and VPO both have a local and global implementation (keyword:*local*). In the local variant, all of the images are treated individually when taking an optimization step, while in the latter the whole band is treated as a single point. A constant 'trust-radius' is used for all optimization methods, where if the magnitude of the maximum Cartesian component of an optimization step exceeds a user-supplied threshold (keyword:*maxmove*), the whole displacement is scaled down accordingly. The number of steps stored in the L-BFGS optimization for the construction of the approximate Hessian matrix can be adjusted by the user (keyword:*lbfgs_memory*). For a more conservative optimization with L-BFGS, the memory can also be erased if a large step is attempted (keyword:*lbfgs_restart_on_maxmove*).

The convergence of the intermediate images is gauged from the maximum Cartesian component of the perpendicular atom force as well as the root-mean-square, i.e. $\max(|\mathbf{F}^{\perp}|), \text{RMS}(\mathbf{F}^{\perp})$. The atomic force on the images perpendicular to the path vanishes as the images are located on the MEP. A typical value of the tolerance for the maximum component of the atomic force perpendicular to the path is $1 \cdot 10^{-3}$ Eh/Bohr. Typically the tolerance for the root-mean-square value is chosen to be smaller by a factor of 1/2 or 1/3. Sometimes a tighter tolerance for the maximum component of the force is needed, for example $5 \cdot 10^{-4}$ Eh/Bohr or even $2 \cdot 10^{-4}$ Eh/Bohr. The maximum number of optimization steps allowed is set by the keyword 'maxiter'.

The configuration of each image after each iteration is written to a '_trj.xyz' file (see file: basename_MEP_ALL_trj.xyz). This file is useful for troubleshooting non-convergent calculations.

4.6.3 Climbing image NEB

In order make the highest energy image converge more accurately to the (highest) energy maximum along the MEP, the climbing image variant of the NEB method (CI-NEB) can be used. [549] In the CI-NEB method, the effective force F^{NEB} acting on the climbing image (i.e. $i = \mathrm{ci}$) is transformed to:

$$F_{\mathrm{ci}}^{\mathrm{NEB}} = F_{\mathrm{ci}} - 2\left(F_{\mathrm{ci}} \cdot \hat{\tau}_{\mathrm{ci}}\right)\hat{\tau}_{\mathrm{ci}}$$

where the climbing image is pushed up-hill along the path and relaxed down-hill perpendicular to the path. That is, the energy is maximized with respect to one degree of freedom corresponding to the direction of the tangent while the energy is minimized with respect to all other degrees of freedom. The effective force on the climbing image does not include any spring force and the density of images then becomes different on either side of the climbing image. As long as the tangent estimate is accurate enough the climbing image will converge rigorously to the point of highest energy along the path. An illustration of how the climbing image NEB method works is shown in Fig. 4.3 for a simple two-dimensional energy function.

Fig. 4.3: Illustration of how the CI-NEB method works on a two-dimensional Müller-Brown energy surface, $E(x, y)$.[556] The calculation is started from a linear interpolation between the reactant (**R**) and product (**P**) energy minima, using $M = 10$. The images are displaced in the orthogonal direction to the path (red curves), until they converge to the minimum energy path (white dashed curve). The climbing image accurately locates the higher energy first-order saddle point along the path (denoted by **SP**). The optimization profile is shown as an inset. In such a plot the interpolated energy along the path is plotted as a function of displacement, for each (or selected) optimization step. The figure is taken from Ref. [553]

It can be useful to start the climbing image after the magnitude of the atomic forces perpendicular to the path drops below a given user supplied threshold (keyword: *tol_turn_on_ci*). Then, the highest energy image along the current path is converted to a climbing image. It is usually most efficient to initiate the climbing image early on or even from the start of the NEB calculation. This applies when using the VPO optimization method. For L-BFGS it is recommended to start CI-NEB when the path has partially converged to the MEP, e.g., around 0.01-0.02 Eh/Bohr.

If there are two or more maxima in the energy along the MEP it is possible that the image near the highest maximum is not chosen as the climbing image at an early stage of the NEB calculation. Then, later on the choice of the climbing image can be switched automatically. Also, for barrierless reactions, the climbing image is not turned on. The atom coordinates of the climbing image (in a CI-NEB calculation) or the highest energy image (in an NEB calculation) are written to files '_NEB-CI_converged.xyz' and '_NEB-HEI_converged.xyz', respectively, when a calculation has successfully completed.

The convergence of a CI-NEB calculation can either be gauged by monitoring the forces on all images or only on the climbing image, (keyword: *convtype*). The latter option may be used when the objective of the calculation is to locate the highest energy saddle point connecting the reactant and product states and can save significant computational effort. To gauge for convergence on the climbing image, both the root-mean-square and magnitude of the maximum (Cartesian) component of the atom force are monitored, i.e. $\max(|\mathbf{F}_{\text{ci}}|), \text{RMS}(\mathbf{F}_{\text{ci}})$. When gauging the convergence of all images in a CI-NEB calculation it is typically acceptable to converge the regular images more loosely than the climbing image. By default, the tolerance for the regular images is a factor of 10 larger than that of the climbing image. This scaling of the tolerances is a parameter that can be set by the user (keyword: *tol_scale*). Typically for an acceptable convergence to the saddle point, the tolerance threshold for the maximum magnitude of an atomic force component of the climbing image can be set to $5 \cdot 10^{-4}$ Eh/Bohr.

4.6.4 Generation of the initial path

One of the most important aspects of any NEB or CI-NEB calculation is the generation of the initial path between the reactant and product energy minima (keyword: *interpolation*. The recommended method is the image-dependent pair potential (IDPP) method [557]. An alternative and simpler method is linear interpolation in Cartesian coordinates. In either case, the user should always inspect the initial path (see file: basename_initial_path_trj.xyz) to make certain that it is acceptable.

By default, the endpoint structures are scanned for covalently linked fragments. If two fragments are detected in either of the endpoint structures, the structure is automatically prepared such that the distance between the fragments is no larger than a maximum distance defaulted to 3.5 Angstrom.

The linear interpolation in Cartesian coordinates may result in overlap of atoms leading to large, initial, atomic forces and even divergence in the SCF cycles. The IDPP method solves these problems by interpolating pairwise distances between neighboring atoms in the reactant and product energy minimum. Then, a path is generated to match those distances as closely as possible. Since there are many more pairwise distances than atom coordinates, the initial path is found by minimizing the objective function,

$$S_{\kappa}^{\text{IDPP}}(r_i) = \sum_{A}^{M} \sum_{B>A}^{M} w(d_{AB}) \left(d_{AB}^{\kappa} - d_{AB}\right)^2 \tag{4.24}$$

where d_{AB} is the pairwise distance between atoms A and B for intermediate image i. d_{AB}^{κ} is the ideal interpolated distance between atoms A and B of the same image. w is a weight function to give shorter bond distances more weight and make unnecessary bond-breaking unfavorable. The weight function is given as $w = (d_{\text{AB}})^{-4}$. The IDPP path avoids the overlap of atoms and can also generate a path that is closer to the MEP than the linear interpolation in Cartesian coordinates[557]. The IDPP path is obtained from an NEB calculation using the IDPP objective function starting from a linear interpolation of the Cartesian coordinates between the endpoint structures, but this calculation requires little computational effort since it does not require any electronic structure computations. Note that it is possible that the initial path breaks covalent bonds and that it can be far from the optimal MEP. If this is the case, the default algorithm detects it and automatically regenerates a different IDPP initial path. This is repeated until either an initial path without unnecessary bond changes is obtained or all initial path option are exhausted, at which point the best initial path is identified. In any case, the user should always inspect the initial path before starting an NEB calculation. The user can adjust the settings of the IDPP calculations using the 'idpp' related keywords, but the default values should suffice for most applications. Note that the units of the IDPP are in Ångströms instead of atomic units. The matrix of ideal interpolated distance between the atoms A and B at image i, $d_{AB,\,i}^{\kappa}$, is most simply obtained by linear interpolation as

$$d_{AB,\,i}^{\kappa} = d_{AB}^{initial} + P_i * \left(d_{AB}^{final} - d_{AB}^{initial}\right) ,$$

where $P_i = \frac{i}{n_{im}-1}$ is the position of image i on the path between 0, corresponding to the initial image, and 1, corresponding to the final image, and n_{im} is the total number of images.

As this linear distance interpolation may not be ideal, one can alternatively choose a bilinear distance interpolation scheme that takes atomic bonds into account. Bonds between the atoms A and B in the initial and final images are identified if their distance $d_{AB} < t_{bond} = t_c \left(R_A^{cov} + R_B^{cov}\right)$, where t_c is an input factor impacting at what distances a bond is detected by default taken to be 1.3, and R_X^{cov} is the covalent radius of atom X. If a bond is detected in one of the endpoint structures but not the other, two linear distance interpolations are performed to obtain $d_{AB,i}^{\kappa}$. The first interpolates the distance between the endpoint structure in which the atoms are bonded and t_{bond}, the second interpolates the distance between t_{bond} and the endpoint structure in which the bond is broken. The two linear distance interpolations are joined at the path position P_{join} defaulted to 0.5, corresponding to the midpoint on the path between the endpoint images. If the atoms are bonded in the initial structure but not in the final structure, the interpolation is

$$d_{AB,\,i}^{\kappa} = \begin{cases} d_{AB}^{initial} + \frac{P_i\left(t_{bond} - d_{AB}^{initial}\right)}{P_{join}}, & \text{if } P_i < P_{join} \\ d_{AB}^{final} - \frac{(1-P_i)\left(d_{AB}^{final} - t_{bond}\right)}{(1.0-P_{join})}, & \text{if } P_i \geq P_{join}\,. \end{cases}$$

If the atoms are bonded in the final structure but not in the initial structure, the interpolation is

$$d_{AB,\,i}^{\kappa} = \begin{cases} d_{AB}^{initial} - \frac{P_i\left(d_{AB}^{initial} - t_{bond}\right)}{1-P_{join}}, & \text{if } P_i < 1 - P_{join} \\ d_{AB}^{final} + \frac{(1-P_i)\left(t_{bond} - d_{AB}^{final}\right)}{P_{join}}, & \text{if } P_i \geq 1 - P_{join}\,. \end{cases}$$

This bilinear interpolation leads to a slower increase of the atom distance in the bonded region compared to the non-bonded region in most cases, emphasizing the bond-breaking process and thus the region where the energy is changed most by a change in the bond distance. If the change in the bond distance in the non-bonded region is slower than the change in the bonded region, the implementation falls back to the regular linear distance interpolation.

In challenging cases, even the IDPP NEB calculation starting from a linear interpolation of the Cartesian coordinates between the endpoint structures may not provide a reasonable initial path. This may happen when the linear interpolation path has atoms come very close to each other in intermediate images, which leads to bond breaking even after NEB optimization using the IDPP objective function. In such cases, a reasonable initial path can often be obtained by avoiding the linear interpolation by sequentially adding images to the path starting from both endpoint structures instead. This method is referred to as sequential IDPP (S-IDPP).[558] The two sets of images close to each endpoint structure are separated by a larger distance than the images in each set by choosing a smaller spring constant than between the images in each set where equal spring constants are used. The images closest to the center are converged in an NEB calculation and an image added at an appropriate distance from the converged image between the two images closest to the center. This process is repeated until the requested amount of images has been added. The tangent of the images closest to the center follows the improved tangent definition and always treats them as extrema on the path, i.e. an average of the normalized distance vectors to the adjacent images is used. The number of images used to form the S-IDPP initial path can be different from the number of images used in the subsequent NEB calculation involving electronic structure calculations. It can be beneficial to use an even number of images in the S-IDPP computation since no image is placed at the center of the initial path then and moving to the requested odd number of images in an additional IDPP NEB calculation afterwards. This option is used by default, but can be deactivated. For systems involving very heavy rotations of large groups, the method becomes more robust when twice the amount of images is used in the S-IDPP calculation. Half the images are then removed automatically for the subsequent NEB calculation involving electronic structure calculations. This option is available, but not used by default.

The initial path generated with the chosen settings is checked for unnecessary bond breaking or bond formation. Bonds between the atoms A and B are identified if their distance $d_{AB} < t_{bond} = t_c \left(R_A^{cov} + R_B^{cov}\right)$, where t_c is an input factor impacting at what distances a bond is detected by default taken to be 1.3, and R_X^{cov} is the covalent radius of atom X. A change in the connectivity of atoms A and B is considered unnecessary if their connectivity does not change between the endpoint structures. If it does change between the endpoint structures, the connectivity is allowed to change exactly once, but any further connectivity changes along the initial path are considered unnecessary. If any unnecessary connectivity changes occur on the initial path, the flexibility of S-IDPP is used to generate more initial paths with different settings systematically, until one without unnecessary connectivity changes is obtained. The parameter space for this hyperparameter search consists of all combinations of the settings *SIDPP_Reparam*, *SIDPP_Energy_Weighted_Tangent*, *SIDPP_Even_NIm, SIDPP_Double_NIm*, *SIDPP_Ideal_Springconst*, and four values (0.05, 0.01, 0.005, and 0.001) for *SIDPP_Tol_MaxF*, leading to a total of 128 options in addition to the chosen settings. Once an initial path without unnecessary connectivity changes is found, the hyperparameter search is terminated. In the unlikely scenario that none of the generated initial paths show no unnecessary connectivity changes, the initial path with the fewest is taken by default, but the user may prefer to have the calculation abort to inspect the initial path, which may be set with the keyword *SIDPP_HyperSearch_StrictConnectivity*.

Note

In certain cases, for instance rotations of large sterically limited groups in a molecule, the reaction may only be possible by breaking and reforming a bond along the MEP. You may not always be looking for an initial path completely free of unnecessary connectivity changes!

The user may have a preconceived notion of the saddle point configuration or have an estimate of the path from a calculation carried out at a lower level of theory. The initial path can be generated in such a way as to include an intermediate configuration as one of the images using the 'NEB_TS_XYZFile' keyword. Since this image will be optimized along with the other intermediate images during the NEB calculation the guess does not have to be accurate.

If inspection of the initial path reveals problems, e.g., unnecessary bond breaking, it is often a good idea to insert a reasonable configuration into the initial interpolation to avoid such problems. Moreover, if an NEB calculation is unable to converge to the MEP (or saddle point) with the given maximum number of iterations, the user can restart the calculation from the 'allxyz' file (see file: basename_MEP.allxyz) which is written to the disk after each iteration during the optimization. Note, when starting an NEB calculation from an output from a previous CI-NEB calculation and vice-versa the band may require a few iterations to adjust the distribution of images to achieve the desired distribution, depending on the selected spring type and the choice of NEB method.

If the system can be modeled reasonably well using the GFN-xTB method, another possible choice is the generation of an initial path on XTB level (keywords 'XTB0', 'XTB1' or 'XTB2' for GFN0-xTB[321] GFN-xTB[320] or GFN2-xTB[161]). In this case the initial path on IDPP level is refined using an NEB calculation on the chosen XTB level. If this NEB run is successful, the entire MEP on XTB level is used as the initial path. If the NEB run on XTB level is not successful, the initial path on IDPP level is used instead.

Another keyword that makes use of the XTB method is the 'XTBTS' keyword ('XTB0TS', 'XTB1TS' or 'XTB2TS'). In this case the initial path on IDPP level is refined using an NEB-CI calculation on XTB level. If the NEB-CI run is successful, the resulting CI structure is chosen as TS guess structure, and the final initial path is generated using an IDPP path from reactant to TS guess and from TS guess to product. If the NEB-CI run on XTB level is not successful, the initial path on IDPP level is used instead.

4.6.5 Removal of translational and rotational degrees of freedom

For NEB and CI-NEB calculations of molecular systems it is important to project out the six (or five in the case of linear molecules) degrees of freedom corresponding to the global rotation and translation of the system. This can be done either at the start of a calculation or for each optimization step (keyword: *quatern*). For the latter, the center-of-geometry of each image is translated to origin and a rotational matrix is constructed using the quaternion approach[559] to minimize the root-mean-square deviation between any two adjacent images. Depending on whether a fixed center is used (keyword: *fix_center*), the images are either kept in that place or transferred back to their original position. This procedure is repeated for any pair of adjacent images in each step of the optimization. Also, the net effective NEB force is set to zero (keyword: *remove_extern_force*).[560]

4.6.6 Reparametrization of the path

In some cases it may be beneficial to enable redistribution of the images along the path every N iterations (i.e. analogous to the string method[561]) (keyword: *reparam*). The path is then interpolated using either a linear or cubic polynomial fitted to both the coordinates and the tangent to the path, and the images are redistributed evenly along the interpolated path. Both N and the type of interpolation are specified by the user (keyword: *reparam_type*). The cubic interpolation method should be better in calculations where the resolution of the path and hence the estimate of the tangent is good, while the linear interpolation is generally more robust as it is not as dependent on the tangent.

4.6.7 Useful output

After each iteration, the energy profile along the path is obtained by making a piecewise cubic polynomial interpolation using both the energy and the component of the atom force tangentinal to the path[546]. The interpolation can reveal important information about the MEP in locations between the intermediate images. The interpolation is written to the file 'interp' (see file: basename.interp) in each step of the optimization. Moreover, as NEB and CI-NEB calculations can be quite computationally demanding and in order to properly analyze what may have gone wrong in such a calculation, the user can inspect the '.log' file, which includes information about the type of calculation, energy, length of the path, spring forces, atomic forces etc.

4.6.8 Important warning messages

Some tests are carried out during the optimization in order to detect problems on the fly. The angle between the two straight lines going through an image and its neighbors on each side is calculated. If the angle becomes large e.g. exceeding 90° the estimate of the tangent has likely become inaccurate and a better resolution of the path is required. If the angle is close to 180° the ordering of the images may have become incorrect. Especially in the latter case, it may be a good choice of action to terminate the calculation and include a larger number of intermediate images in the subsequent calculation. Some information from the calculation, e.g, a guess for the saddle point, could be incorporated in the new calculation.

Another issue is the identification of an intermediate minimum along the path. If an intermediate minimum is observed in M subsequent iterations (M is supplied by the user) a warning is issued to the user that a possible intermediate minimum may exist along the path. It is a good idea to check the status of the calculation, the path and the convergence behaviour. If the calculation appears to be proceeding normally and heading for convergence, the best course of action is to allow the calculation to finish. However, it is in general better to carry out separate NEB calculations for segments of the MEP on either side of the intermediate minimum. Especially, if the intermediate minimum is deep w.r.t to the reactant and product state energy minima.

In such cases, the image closest to the apparent intermediate minimum is selected and a structural minimization is carried out. The resulting configuration is then used as the initial state or final state in the subsequent CI-NEB calculations.

4.6.9 Parallel execution

If the number of processes (NProcs) specified in the input is larger than 1, NEB will automatically start up in multi-process mode:

NProcs <= NImages
: NProcs processes will handle NProcs images independently with 1 process per image. Choose NProcs = X*NImages (e.g. X = 1 or 0.5)

NProcs > NImages
: NProcs processes will handle NImage images, each image being treated by (NProcs/ NImages) processes. If you want to dedicate more than 1 process to each single image-calculation, choose NProcs = X*NImages (e.g. X = 2, 3, 4, …).

Note: If in the second case multiple compute-nodes are involved, the user will need to define the ORCA-specific environment variable RSH_COMMAND, which tells the NEB driver how to connect to the individual nodes (set it to either 'rsh' or 'ssh'). However, this may not work with all queueing systems.

If the energy and force calculations are fast (e.g. with semiempirical methods), there is no gain in using multiple processes per image. Starting up and finalizing MPI may consume more time than the gain from parallel processing.

4.6.10 zoomNEB

A preliminary version of the Zoom-NEB (Z-NEB) method has been included in this implementation, where the objective of the method is to locate a saddle point more accurately with a better resolution compared to CI-NEB calculations. The Z-NEB method is an automatic two-step procedure, where in the first step a CI-NEB calculation is carried out to obtain a rough convergence towards the MEP. Then, the region surrounding the highest energy maximum along the path is identified and a new set of images is distributed along this region. In the second step, a free-end CI-NEB calculation is started from this new path. In this calculation, the endpoints are optimized according to the atom force acting perpendicular to the path. This will ensure that the endpoints of the second CI-NEB calculation will converge to the MEP, as well. Furthermore, to maintain the parallelization of the CI-NEB method, the same number of movable images are used in the first and second CI-NEB calculations.

4.6.11 NEB-TS

Probably the most efficient saddle point search methods are obtained when double-ended methods (e.g. NEB) are combined with single-ended methods (e.g., eigenvector-following). In the current implementation, a combination of EW-CI-NEB and EF P-RFO (eigenvector-following partitioned rational function optimization) methods is presented and referred to as the NEB-TS method [543].

In NEB-TS, the EW-CI-NEB method is used to partially converge to the MEP and hence saddle point, i.e., the optimization of the images along the MEP is halted once the climbing image is converged to a prescribed tolerance. Then, the climbing image is used to provide a subsequent eigenvector-following calculation with an accurate initial guess configuration, as is shown in Fig. 4.4 and the tangent at the CI is used to select the eigenvector to be followed. The tangent estimate should already provide an accurate approximation to the unstable mode at the saddle point. The initial Hessian matrix needed for the eigenvector-following calculation can either be computed analytically (if available) or taken as the Almlöf empirical Hessian matrix [8]. If the Almlöf Hessian matrix is used, the curvature at the CI is estimated by using a finite difference approximation (i.e. using the atom force acting on the neighboring images to CI) and used to scale the corresponding eigenvalue of the selected eigenvector, allowing the eigenvector-following calculation to be started from a Hessian matrix of the correct form. The typical '%geom' block can be used to modify the settings of the eigenvector-following calculation.

Fig. 4.4: Illustration of how the NEB-TS method works on a two-dimensional Müller-Brown energy surface, $E(x, y)$ [556]. The calculation is started from a linear interpolation between the reactant and product energy minima, using $M = 10$. The images are displaced in the orthogonal direction to the path (red curves) using the CI-NEB algorithm, until a rough convergence to the minimum energy path (white dashed curve) is obtained. The climbing image then provides an approximate saddle point configuration that can be used to start eigenvector-following partitioned rational function optimization to accurately (and swiftly) identify the (higher energy) first-order saddle point. The figure is taken from Ref. [543]

4.6.12 FAST-NEB-TS and LOOSE-NEB-TS

Since our first NEB-TS implementation, we investigated a lot more settings and variants, see [543]. Based on those findings, two new algorithms and convergence threshold settings have been implemented into ORCA. FAST-NEB-TS corresponds to the IDPP-TS in the paper, and LOOSE-NEB-TS corresponds to the actual NEB-TS defaults, which are defined in the paper. Both features, FAST- and LOOSE-NEB-TS, show slightly lower robustness, but need significantly fewer NEB cycles.

4.6.13 NEB / NEB-TS and TD-DFT

The NEB and NEB-TS algorithm now also works in combination with TD-DFT. The input:

```
! NEB-TS
%neb
 product "product.xyz"
end
%tddft
 NRoots 1
 IRoot 1
end
```

not only computes the MEP and TS of the first excited state, but it also prints out (after NEB convergence) the excited state as well as ground state energies over the MEP:

```
----------------------------------------
Image    E0      Root  1
----------------------------------------
  0   -67.203     0.000
  1   -66.443     1.517
  2   -54.382     5.774
  3   -42.483    10.473
  4   -32.798    14.323
  5   -28.056    16.108
  6   -34.102    13.915
  7   -48.125     8.249
  8   -64.188     2.212
  9   -67.066     0.018
```

You can even request an NEB-TS calculation on the ground state PES, and at the same time gain information on the excited PES via the input:

```
! NEB-TS
%neb
 product "product.xyz"
end
%tddft
 NRoots 1
 IRoot 0
end
```

During this NEB calculation, only ground state energies and gradients are computed. Only after NEB convergence, the additional excited state energies are computed on each image, in order to yield the ordering of the states on the MEP:

```
----------------------------------------
Image    E0      Root  1
----------------------------------------
  0     0.000   121.545
  1     6.281   128.198
  2    21.434   143.835
  3    35.437   151.581
  4    43.223   155.579
  5    43.235   155.501
  6    35.472   151.495
  7    21.498   143.715
  8     6.395   128.060
  9     0.166   121.425
```

4.6.14 Summary of Keywords

The following keywords are available:

```
! NEB            # NEB calculation
! NEB-CI         # Climbing Image NEB calculation
! NEB-TS         # NEB calculation plus subsequent TS optimization
! FAST-NEB-TS    # NEB calculation with one iteration only plus subsequent TS opt.
! LOOSE-NEB-TS   # NEB calculation with default convergence criteria from NEB-TS␣
↪paper
! TIGHT-NEB-TS   # NEB calculation with tighter convergence criteria plus
                 # subsequent TS optimization
! ZOOM-NEB       # NEB calculation plus zoomed NEB calculation
! ZOOM-NEB-CI    # Climbing Image NEB plus zoomed climbing image NEB calculation
! ZOOM-NEB-TS    # NEB calculation plus zoomed NEB calculation plus subsequent
                 #  TS optimization
```

(continues on next page)

(continued from previous page)

```
! NEB-IDPP      # IDPP (Initial Path) NEB calculation - for estimation of path
                #  length
%neb
   Product "product.xyz" # product structure. Input is mandatory.
   NImages  8         # default 8. Number of images without fixed endpoints,
                      #  for free_end total number of images
   PrintLevel   1     # default 1. Normal printout. Use 0 for no printout, higher
                      #  numbers (<=4) for more detailed printout.

   TS "TSGuess.xyz"   # Provide guess for the TS structure. Images
                                  #  are interpolated between reactant and TS guess
                                  #  and between TS guess and product.
   NEB_TS_Image     3 # default -1. Number of the image the TS guess is used for.
                      #  If not defined (=-1), the image which gives lowest RMSD
                      #  for all image distances is used.

 # Restart option: After each iteration the NEB method stores all image
 # structures in an .allxyz file. In case of an abort this file can be used
 # for a restart. File should contain the structures for all images.
   Restart_ALLXYZFile "NEB1.allxyz"# use the trajectory from file if filename is
                                   #  provided

 # Alternatively, NEB can be started on user-prepared wavefunctions for each image.
 # The names of these wavefunction files should consist of a user-chosen basename
 # and the extension '_imN.gbw', where N is the image number.
 # The basename should be provided in the input, ORCA will add extension '_imN.gbw'
   Restart_GBW_BaseName "NEB2"     # use the wavefunctions from file NEB2_imN.gbw

 # Check SCF convergence: If true, SCF convergence is checked for and
 # calculation aborts if:
 # -any of the images does not show SCF convergence in four subsequent cycles.
 # -any of the images does not show SCF convergence in two subsequent cycles
 #  after the gradient is converged.
   CheckSCFConv     true           # default true

 # PDB file input format:
   Product_PDBFile  "product.pdb"  # Product structure in pdb format. If this is
                                   #  given, xyz does not need to be given.
   TS_PDBFile       "TSGuess.pdb"  # TS guess structure in pdb format. If this is
                                   #  given, xyz does not need to be given.
   Free_End         false          # Use free-end NEB. In this case the NImages
                                   #  corresponds to the total number of images.
   PreOpt           false          # do optimization of reactant and product in
                                   #  internal coordinates before NEB starts
   NSteps_FoundIntermediate  30    # Number of steps the intermediate has to be
                                   #  present
   AbortIf_FoundIntermediate false # If an intermediate is found abort the run.
   NPTS_Interpol    10             # Number of abscissa in cubic polynomial
                                   #  interpolation
   Interpolation    IDPP           # Method to generate the images based on the
                                   #  reactant, product (and potentially TS guess)
                                   # linear
                                   # IDPP
                                   # XTB0TS  - TS on GFN0-xTB level
                                   # XTB0    - entire path on GFN0-xTB level
                                   # XTB1TS  - TS on GFN1-xTB level
                                   # XTB1    - entire path on GFN1-xTB level
                                   # XTB2TS   - TS on GFN2-xTB level
                                   # XTB2     - entire path on GFN2-xTB level
   Prepare_Frags    true           # Analyze endpoint structures for fragments.
                                   #  If two fragments are detected in an
```

(continues on next page)

(continued from previous page)

```
                                  #  endpoint structure, reduce distance to
                                  #  maximum distance given by Max_Frag_Dist
  Max_Frag_Dist    3.5            # Maximum allowed fragment distance. If they
                                  #  are farther apart, reduce the distance to
                                  #  this value (Ang.)
  Bond_Cutoff      1.2            # Factor to multiply sum of covalent radii of
                                  #  two atoms by. If the distance is smaller
                                  #  than the result, the atoms are considered
                                  #  bonded.

 # The formulation used to estimate the tangent to the path
  Tangent improved                # improved (default)
                                  # original

 # The type of the spring interaction parallel to the path. Original springs apply
 # spring interaction between each degree of freedom of adjacent images, while
 # 'image' springs apply a spring interaction between the images
 # Spring type
  SpringType image                # image / distance (default)
                                  # dof / original
                                  # ideal
  SpringConst  0.01               # The spring constant used to scale the spring
                                  #  forces parallel to the path. If energy-
↪weighted
                                  #  springs are used. This parameter gives the
                                  #  lower bound value of the spring constant
  SpringConst2 0.1                # If energy-weighted spring forces are used.
                                  #  This parameters give the value for the upper
                                  #  bound value of the spring constant.
  Energy_Weighted true            # Employ energy-weighted springs. When
                                  #  energy-weighted springs are used, the
                                  #  images tend to accumlate in higher energy
                                  #  regions of the path.

 # The type of the spring interaction perpendicular to the path. The perpendicular
 # spring is introduced via a scaling function: cos, tan, costan, which all use
 # the spring component perpendicular to the path.
 # DNEB is the doubly nudged elastic band method.
  PerpSpring    no                # no      (default)
                                  # cos
                                  # tan
                                  # cosTan
                                  # DNEB

   LLT_Cos true                   # Enables the cos-type spring force
                                  #  acting perpendicular to the band.

 # Translational and rotational degrees of freedom
 Quatern     always           # no,
                              # startonly
                              # always (default)
 # Fix_center specifies whether the centroid of each image should be
 # constrained to the origin of the coordinate system or to the center
 # of each image individually.
 Fix_center  True

 # Fix_center specifies whether the centroid of each image should be
 # constrained to the origin of the coordinate system or to the center
 # of each image individually.
 Remove_extern_Force  True    # Removes the net effective NEB force before
                              #  translation of the path
```

(continues on next page)

(continued from previous page)

```
 # Options for Free-End NEB
  Free_End_Type    Perp       # Type of optimization of endpoints in free-end
                              #  NEB.
                              # contour - constrain endpoints to a fixed
                              #  contour with energy EC, see below
                              # perp - allow end points to move according to
                              #  perp. spring force
                              # full - allow to move according to full force,
                              #  i.e. relax to energy minimum
  Free_End_EC                 # Energy contour value for image 0 - needed for
                              #  free_end_type = contour
  Free_End_EC_End             # Energy contour value for image N - needed for
                              #  free_end_type = contour
  Free_End_Kappa              # harmonic restraint term - needed for
                              #  free_end_type = contour


# Monitor convergence for all images or only the CI.
# Convergence type
  ConvType all                     # all   (default)
                                   # CIOnly

  CI false                         # Do Climbing image NEB

  NEB_TS false                     # Do CI NEB and subsequent TS opt.

# Convergence tolerance. In Eh / Bohr (except Tol_Scale ).
  Tol_MaxFP_I      1.e-3  # Default. The convergence tolerance for the
                          #  maximum component of the atomic force
                          #  perpendicular to the path.
  Tol_RMSFP_I      5.e-4  # Default. The convergence tolerance for the rms
                          #  atomic force perpendicular to the path. Only
                          #  applies to regular images.
  Tol_MaxF_CI      2.e-3  # The convergence tolerance for the maxmimum
                          #  component of the atomic force acting on the CI.
                          #  Only applies to (ZOOM-)NEB-CI/-TS calculations.
                          #  Default is 5.e-4 (-CI) and 2.e-3 (-TS)
  Tol_RMSF_CI      1.e-3  # The convergence tolerance for the rms atomic
                          #  force acting on the CI. Only applies to (ZOOM-)NEB-
↪CI.
                          #  Default is 2.5e-4 (-CI) and 1.e-3 (-TS)
  Tol_Turn_On_CI   2.e-2  # Thresholds for max. atomic force for switching on
                          #  CI in (ZOOM-)NEB-CI and (ZOOM-)NEB-TS.
                          #  Defaults: 0.02 for LBFGS, 0.2 for VPO and FIRE
  Tol_Scale        10.0   # For convergence type 'all' the user can scale
                          #  the convergence tolerance of the regular images
                          #  relative to the CI values using this
                          #  multiplicative factor. Only applies to (ZOOM-)NEB-CI
                          #  and (ZOOM-)NEB-TS calculations.

 # Interpolation and redistribution of the path is performed every 'reparam'
 # iterations. The type of interpolation is set by reparam_type.
  Reparam_type linear   # Cubic
                        # Linear (default)
  Reparam 0             # No. of iterations after which the path should be
                        #  reparametrized
                        #  0 (default) means: reparametrization is off
  Tol_Reparam 0.0       # User-defined threshold at which the path should be
                        #  reparametrized
                        # 0.0 (default) means: reparametrization is off
```

(continues on next page)

(continued from previous page)

```
 # The optimization method used to converge the band on the MEP / saddle point.
 # The L-BFGS is more aggressive and efficient, but also more error-prone.
 # VPO is conservative and robust.
  Opt_Method LBFGS      # LBFGS   (default)
                        # VPO
                        # FIRE
                        # BFGS

# Options Optim. Method
  Maxmove  0.1    # maximum component allowed per step. Default is 0.1 (LBFGS)
                  #  and 0.2 (VPO / FIRE)
  Stepsize 1.0    # multiplicative factor to scale the size of the step in each
                  # optimization cycle.
                  #  Default is 1.0 (LBFGS) and 0.5 (VPO / FIRE)
  MaxIter  500    # Maximum number of iterations. 500 for LBFGS, 1000 for VPO /␣
↪FIRE.
  Local    false  # Use local optimization.
                  #  Default is false for NEB, but true for (ZOOM-)NEB-CI/-TS.

# Options LBFGS
  LBFGS_Mem      20    # the number of previous steps to be kept in memory and␣
↪used
                       #  to construct the approximate Hessian matrix.
  LBFGS_DR 1.e-3       # Size of the finite difference step taken at the
                       #  initialization of L-BFGS
  LBFGS_Restart_On_Maxmove true  # Re-initialize L-BFGS for the next step when
                                 #  the 'max-move' limit is reached.
  LBFGS_Reparam_On_Restart false # Re-parametrize when L-BFGS is re-initialized
  LBFGS_Precondition true        # If true, then after initialization, the␣
↪curvature
                                 #  along direction of the force is estimated and
                                 #  used to determine the first step

# FIRE parameters
  FIRE_INITIAL_DAMP 0.1   # Initial value for the damping factor
  FIRE_DAMP_DECR"   0.99  # Decrease of the damping factor
  FIRE_STEP_INCR"   1.1   # Factor to increase the stepsize
  FIRE_STEP_DECR"   0.5   # Factor to decrease the stepsize
  FIRE_MAX_STEP"    5.0   # Default is 10 x Stepsize
  FIRE_RETENTION"   5     # Retention before starting acceleration

# Options Zoom
  Tol_Turn_On_Zoom  0.1   # use ZOOM-NEB(-CI/TS)
  Zoom_Offset       1     # if manual selection is chosen, how many
                          #  images away from CI should be chosen
  Zoom_Auto         true  # automatically select zoom region
  Zoom_Alpha        0.5   # determines how much of the barrier
                          #  zoom-auto should select
  Zoom_Interpolation      # linear (default)
                          # cubic
  Zoom_PrintFullTrj       # print full trajectory including fixed region during␣
↪Zoom

# Set of parameters to adjust the IDPP pre-optimization when generating the␣
↪initial
# path.
# Options IDPP
  IDPP_NMax                     7000  # maximum number of cycles allowed in IDPP
  IDPP_Tol_MaxF                 0.01  # tolerance on the maximum component of the
                                      #  atomic force perpendicular to the path.
                                      #  For S-IDPP, this setting is used in the
```

(continues on next page)

(continued from previous page)

```
                                        #  final IDPP optimization after all images
                                        #  have been added to the path
   IDPP_ksp                      1.0   # spring constant used to scale the spring
                                        #  force parallel to the path.
   IDPP_Alpha                    0.01  # multiplicative factor to scale the size
                                        #  of the step in each opt. cycle
   IDPP_MaxMove                  0.05  # maximum component allowed per step
   IDPP_Debug                    false # will print out the convergence of IDPP
                                        #  and also the optimization trajectory and
                                        #  the log file for the IDPP run.
   IDPP_Quatern                  true  # Whether quaternions should be used in the
                                        #  IDPP optimization
  # Interpolation scheme between the endpoint images to use for the pairwise atom
  # distances to obtain the ideal distances. Bilinear identifies bonded atoms in
  # one endpoint structure that are not bonded in the other and leads to a smaller
  # change in the bond distance in the bonded region than in the non-bonded one.
   IDPP_Dist_Interpolation             # Linear (default)
                                        # Bilinear
   IDPP_Bilinear_Partition       0.5   # Path position at which to join the two
                                        # linear distance interpolations. The␣
↪default
                                        # corresponds to the midpoint of the path.
   SIDPP                         false # Whether the IDPP optimization should use a
                                        #  Cartesian linear interpolation initial
                                        #  guess or sequentially add images to the
                                        #  path
   SIDPP_Tol_MaxF                0.01  # Tolerance on the maximum component of the
                                        #  atomic force perpendicular to the path
                                        #  acting on the images closest to the␣
↪center
                                        #  of the path. This setting defines when
                                        #  new images are added to the path
   SIDPP_Reparam                 true  # Whether to reparameterize the path once
                                        #  after all images have been added to it
                                        #  (uses Reparam_type setting) before the
                                        #  final IDPP optimization of the full path
   SIDPP_Energy_Weighted_Tangent false # Whether to use an energy weight in the
                                        #  tangent definition of the images closest
                                        #  to the center of the path
   SIDPP_Even_NIm                true  # Whether to perform S-IDPP with one image
                                        #  less than requested until all images
                                        #  have been added to the path and then
                                        #  add the last image afterwards. This
                                        #  setting increases robustness of the
                                        #  method since no image is placed in the
                                        #  center of the path during image addition
   SIDPP_Double_NIm              false # Whether to perform S-IDPP with twice the
                                        #  amount of images and then remove half of
                                        #  them automatically. This setting
                                        #  increases robustness of the method for
                                        #  systems involving very heavy rotations of
                                        #  large groups
   SIDPP_Ideal_Springconst       false # Whether the spring constant between the
                                        #  images closest to the center of the path
                                        #  should be scaled according to the ratio
                                        #  of the number of images that have been
                                        #  placed on the path already and the
                                        #  requested number of images. This is a
                                        #  more aggressive setting pulling the
                                        #  images closest to the center of the path
                                        #  together more.
```

(continues on next page)

(continued from previous page)

```
  SIDPP_HyperSearch            true  # Check connectivity of initial path
                                     #  and perform hypersearch for S-IDPP
                                     #  initial path with minimal
                                     #  unnecessary connectivity changes?

  SIDPP_HyperSearch_StrictConnectivity
                               false # Abort if no initial path with no
                                     #  unnecessary connectivity changes
                                     #  is found? Otherwise, use the one
                                     #  with the fewest

# Extra Output options
  Monitor_Internals # Up to three internal coordinates can be defined
     {B 0 1}        # for which the values are printed during the NEB run.
     {B 1 5}        #  Possible are (B)onds, (A)ngles, (D)ihedrals and (I)mpropers
  end
end
```

Output files:

- Configuration and trajectory files:
  - basename_initial_path_trj.xyz: The initial path generated at the start of the NEB run and after minimization of RMSD between the reactant and product states.
  - basename_MEP_trj.xyz: The final converged MEP trajectory.
  - basename_MEP_ALL_trj.xyz: The configurations of each image are appended to this file for each step of the NEB optimization.
  - basename_trj.xyz: The trajectory of TS optimization.
  - basename_MEP.allxyz: Restart file that includes the configuration of each image from the last iteration of an NEB or NEB-CI iteration.
  - basename_NEB-CI_converged.xyz: The configuration of the climbing image after a successful NEB-CI calculation.
  - basename_NEB-HEI_converged.xyz: The configuration of the highest energy image after a successful NEB calculation.
  - basename.xyz: The configuration of the optimized saddle point using the TS optimization.
- Log files:
  - basename.interp: The interpolated energy profile of the path for each iteration during the NEB/NEB-CI optimization.
  - basename.interp.final: The energy profile for the converged path of an NEB/NEB-CI optimization.
  - basename.log: A general log file containing essential information regarding the run e.g., energy, forces and step size.

4.7 Vibrational Frequencies

Vibrational frequency calculations are available through analytical differentiation of the SCF energy as well as one- or two-sided numerical differentiation of analytical gradients, i.e. for Hartree-Fock and DFT models. For methods without analytical gradient, a numerically calculated gradient can be used (keyword `NumGrad`) for numerical frequencies. Please note that this will be a very time-consuming calculation.

At present, analytical Hessians can be calculated for SCF only. However, there are some additional restrictions. Analytical Hessians cannot be performed for

- Double-Hybrid functionals
- RI-JK approximation

Note that there is a separate section of the manual dedicated to vibrational spectroscopy (see section *Vibrational Spectroscopy*) that contains further examples and describes the calculation of anharmonic effects, Raman spectra, NRVS and things like isotope shifts or visualizing normal modes.

The use of vibrational frequency calculations is fairly simple:

```
# any Hartree-Fock or DFT model can be used here
! BP def2-TZVP

# Tight SCF convergence is advisable to minimize the numerical
# noise in the frequencies.
! TightSCF

# perform a geometry optimization first
! Opt

# Run an analytical or numerical frequency calculation afterwards
! AnFreq   # or just ``! Freq''
# numerical:
! NumFreq

# details of the numerical frequency calculation
%freq  CentralDiff true   # use central-differences (this is the default)
       Increment   0.005  # increment in bohr for the
                          # differentiation (default 0.005)
       end

! bohrs
* xyz 0 1
 O          -1.396288    -0.075107     0.052125
 O           1.396289    -0.016261    -0.089970
 H          -1.775703     1.309756    -1.111179
 H           1.775687     0.140443     1.711854
*
```

At the end of the frequency job you get an output like this:

```
-----------------------
VIBRATIONAL FREQUENCIES
-----------------------

   0:         0.00 cm**-1
   1:         0.00 cm**-1
   2:         0.00 cm**-1
   3:         0.00 cm**-1
```

(continues on next page)

(continued from previous page)

```
  4:          0.00 cm**-1
  5:          0.00 cm**-1
  6:        311.78 cm**-1
  7:        887.65 cm**-1
  8:       1225.38 cm**-1
  9:       1394.81 cm**-1
 10:       3624.88 cm**-1
 11:       3635.73 cm**-1
```

This output consists of the calculated vibrational frequencies, the vibrational modes and the thermochemical properties at 298.15 K. In the example above there are six frequencies which are identically zero. These frequencies correspond to the rotations and translations of the molecule. They have been projected out of the Hessian before the calculation of the frequencies and thus, the zero values do not tell you anything about the quality of the Hessian that has been diagonalized. The projection can be turned off by `PROJECTTR FALSE` under %FREQ, so that the frequencies of the translations and rotations can deviate from zero and the deviations represent a metric of the numerical error of the Hessian calculation. This is done automatically when there is e.g. an external electric field that makes the exact translational and/or rotational modes have non-zero frequencies (see section *Finite Electric Fields*). However, in normal cases where the molecule is expected to obey both translational and rotational invariance, it is **strongly discouraged** to turn off `PROJECTTR` when calculating thermochemical quantities (especially entropies and Gibbs free energies). This is because when the frequencies of translational and rotational modes exceed `CutOffFreq` (which is 1 cm^{-1} by default), their contributions to the partition function will be calculated using the formulas for vibrations. As a result, the calculated entropy is inaccurate (due to treating translations and rotations as vibrations), is sensitive to numerical noise, and in particular exhibits a finite jump when the (theoretically zero) frequencies of the translational and rotational modes cross `CutOffFreq`. Therefore, the only case where the user needs to turn off `PROJECTTR` manually is when the exact Hessian is expected to have zero translational and rotational frequencies, and one wants to check how much the translational and rotational eigenvalues of the actually computed Hessian deviate from zero. The thermochemical quantities from such a calculation are less reliable and should not be used; even if they differ considerably from the results with `PROJECTTR TRUE`, this does not necessarily mean that the latter are unreliable.

Without `PROJECTTR FALSE`, the reliability of the calculated frequencies has to be judged by comparison of calculations with different convergence criteria, increments, integration grids etc. The numerical error in the frequencies may reach 50 cm^{-1} but should be considerably smaller in most cases. Significant negative frequencies indicate saddle points of the energy hypersurface and prove that the optimization has not resulted in an energy minimum.

OBS: By default, the Hessian is made translation invariant by applying the "acoustic sum rule" ([562]), which reduces the effect of noise from numerical integration coming from DFT or COSX, except for the Partial and Hybrid Hessians where it does not make sense. It can be set to false by using TRANSINVAR FALSE under %FREQ.

4.7.1 Mass dependencies

Of course the calculated frequencies depend on the masses used for each atom. While this can be influenced later through the `orca_vib` routine (see Section *Isotope Shifts* for more detail) and individually for each atom in the geometry input, one might prefer using a set of precise atomic masses rather than the set of atomic weights (which are set as default). This can be achieved through the `!Mass2016` keyword, which triggers Orca to use those atomic masses representing either the most abundant isotope or the most stable isotope (if all isotopes are unstable) of a certain element (e.g. the mass of ^{35}Cl for chlorine or the mass of ^{98}Tc).

Note

The calculation of numerical frequencies puts rather high demands on both computer time and accuracy. In order to get reliable frequencies make sure that:

- Your SCF is tightly converged. A convergence accuracy of at least 10^{-7} Eh in the total energy and 10^{-6} in the density is desirable.
- Grids of at least `DEFGRID2` (default) are used.

- The use of two-sided (i.e. central) differences increases the computation time by a factor of two but gives more accurate and reliable results.
- Small auxiliary basis sets like DGauss/J or DeMon/J may not result in fully converged frequencies (up to 40 cm^{-1} difference compared to frequencies calculated without RI). The def2/J universal auxiliary basis sets of Weigend that are now the default in ORCA (or the SARC/J for scalar relativistic calculations) are thought to give sufficiently reliable results.
- Possibly, the convergence criteria of the geometry optimization need to be tightened in order to get fully converged results.
- If you can afford it, decrease the numerical increment to 0.001 Bohr or so. This puts even higher demands on the convergence characteristics of the SCF calculation but should also give more accurate numerical second derivatives. If the increment is too small or too high inaccurate results are expected.

The calculation of analytical frequencies is memory-consuming. To control memory consumption the `%maxcore` parameter must be set. For example, `%maxcore 8192` - use 8 Gb of memory per processor for the calculation. The user should provide the value according to the computer available memory. The batching based on `%maxcore` parameter will be introduced automatically to overcome probable memory shortage.

Numerical frequency calculations are restartable (but analytical frequency calculations are not). If the numerical frequencies job died for one reason or another you can simply continue from where it stopped as in the following example:

```
! STO-3G NumFreq
%freq  Restart true  # restart an old calculation
                     # this requires .res.* files to be present
       end
* int 0 1
    C  0 0 0 0.0000   0 0
    C  1 0 0 1.2160   0 0
    H  1 2 0 1.083  180 0
    H  2 1 3 1.083  180 0
*
```

Note

- You must not change the level of theory, basis set or any other detail of the calculation. Any change will produce an inconsistent, essentially meaningless Hessian.
- The geometry at which the Hessian is calculated must be identical. If you followed a geometry optimization by a frequency run then you must restart the numerical frequency calculation from the optimized geometry.
- Numerical frequencies can be performed in multi-process mode. Please see section *Parallel and Multi-Process Modules* ("Hints on the use of parallel ORCA") for more information.
- The restart of Numerical frequencies will take off from the result files produced during the preceding run (`BaseName.res.%5d.Type`, whith `Type` being `Dipoles`, `Gradients` - and if requested `Ramans` or `Nacmes`). Please make sure that all these local result files get copied to your compute directory. If restart is set and no local files to be found, ORCA will restart from scratch. If ORCA finds a Hessian file on disk, it will only repeat the subsequent analysis.
- The Hessian can be transformed to redundant internal coordinates. More information can be found in section *Printing Hessian in Internal Coordinates*.

4.7.2 Scaling factors for computed frequencies

When comparing computed vibrational frequencies to experiment, effects that are not regarded in the harmonic approximation are often corrected for by scaling the computed harmonic freuquencies.

First of all, a standard frequency calculation has to be carried out. After that, a .hess file can be found on disk. This file contains a line with the frequency scaling factor, that can be changed manually:

```
$frequency_scale_factor
  0.90      <<<---- you change this to whatever you want
```

then, running `orca_vib` is required:

```
orca_vib  myjob_scaled_freq.hess
```

The program reads the Hessian, diagonalizes it and applies the scaling factor. Whatever scaling factor was used in the actual input that generated the Hessian is irrelevant since the Hessian is re-diagonalized. To avoid confusion, we recommend that if the goal is to play with the scaling factor, leave the scaling factor in the input at 1.0. Nothing bad happens if you don't though.

4.7.3 Restarting Numerical Frequency calculations

To restart a numerical frequencies calculation, use:

```
%FREQ
  restart true
END
```

and ORCA will look for basename.res.{} files in the same folder where the calculation is being run, check for what has already been done and restart where it is needed.

4.7.4 Frequencies keyword list

In the ORCA program package, the calculation of frequencies through the numerical or analytical Hessian is done via the `orca_numfreq` module and the combination of the `orca_propint`, `orca_scfresp`, and `orca_prop` modules, respectively.

The parameters to control these frequency calculations can be specified in the %freq-block. Relevant simple input keywords are given in Table 4.6.

```
%freq

  # Flags to switch frequencies calculation on/off

  NumFreq    false       # numerical frequencies (available for all methods)
  AnFreq     false       # analytical frequencies (available for HF, DFT)
                         # (One of these options has to be set to true,
                         #  to request a freq calculation)

  ScalFreq   1.0         # Scaling factor for frequencies (default = 1.0)
                         # NOTE: Scaling is applied to the frequencies after they␣
↪are
                         #       calculated. SCALED frequencies will be stored in␣
↪the
                         #       .hess file and printed in the output file.
                         #       In the .hess file you have accesss to the␣
↪frequency
                         #       scaling factor (see below).
```

(continues on next page)

(continued from previous page)

```
  # Flags to control NumFreq calculation:

  CentralDiff  true      # use central differences [f(x+h)-f(x-h)]/2h - or -
                         # use one-sided differences [f(x+h)-f(x)]/h
  Restart      false     # restart a (numerical) frequency calculation
  DX           0.005     # increment h
  Increment    0.005     # increment h

  Hybrid_Hess  {...} end # calculate (numerical) Hybrid Hessian
  Partial_Hess [...] end # calculate (numerical) Partial Hessian


  # Flags to control subsequent vibrational analysis:

  QuasiRRHO       true   # Evaluate Vibrational Entropy with
                         # Quasi-Rigid Rotor Harmonic Oscillator
  QRRHORefFreq    25     # reference frequency used in the QuasiRRHO in cm-1
                         # default value is 100 from original paper
  CutOffFreq      1.0    # Threshold for frequencies to be considered
                         # in spectra, thermochemistry and printout (cm-1)
  Temp        298.15     # run the thermochemistry calculations at user-defined
                         # temperatures (max 16 temperatures, separated by ',')
  T    290, 292, 295     # same as Temp

  XTBVPT2     True       # use XTB for the VPT2 correction of the IR
  Delq           0.5     # the displacement in dimensionless coordinates used
                         # during the VPT2

  TransInvar  True       # enforce translation invariance after calculating the␣
↪Hessian?
  NumHessTransInvar True # enforce translation invariance while calculating the␣
↪Hessian?
                         # NOTE: TransInvar does not reduce the computational cost␣
↪of the
                         # Hessian, and can be applied to analytic as well as␣
↪numeric
                         # Hessians. NumHessTransInvar reduces the number of␣
↪gradients
                         # that need to be calculated by 6 (for two-sided␣
↪differentiation)
                         # or 3 (for one-sided differentiation) for numeric␣
↪Hessians, but
                         # has no effect for analytic Hessians.
  ProjectTR   True       # project out translation and rotation degrees of freedom
                         # in frequency calculation and thermochemistry analysis?



end
```

Table 4.6: Simple input keywords for vibrational frequency calculation

Keyword	Description
`Freq`	Perform an analytic frequency calculation (Alias: `AnFreq`)
`NumFreq`	Perform a numerical frequency calculation
`Mass2016`	Use the latest (2016) atomic masses of the most abundant or most stable isotopes instead of atomic weights

4.8 Thermochemistry

The second thing that you get automatically as the result of a frequency calculation is a thermochemical analysis based on ideal gas statistical mechanics. This can be used to study heats of formation, dissociation energies and similar thermochemical properties. To correct for the breakdown of the harmonic oscillator approximation for low frequencies, entropic contributions to the free energies are computed, by default, using the Quasi-RRHO approach of Grimme.[280] To switch-off the Quasi-RRHO method and use the RRHO method, use:

```
%freq  QuasiRRHO false
       CutOffFreq 35 # in cm-1
       end
```

Where the CutOffFreq parameter controls the cut-off for the low frequencies mode (excluded from the calculation of the thermochemical properties). Note that the default CutOffFreq is 1 (cm^{-1}) when Quasi-RRHO is on, since Quasi-RRHO behaves much more reasonably for low frequencies than RRHO does. In particular, the entropy contribution calculated by Quasi-RRHO approaches a constant value when the vibrational frequency approaches zero, while the RRHO contribution diverges.

The Quasi-RRHO method smoothly interpolates between the entropy formulas of a harmonic oscillator and a rigid rotor, such that high frequency vibrations behave like harmonic vibrations, and low frequency vibrations behave like rotations with the same frequency. The frequency at which the entropy contribution is a half-half mixture of rotation and vibration is called the "reference frequency" ω_0 of the Quasi-RRHO method, accessible via the `QRRHORefFreq` keyword in `%freq` (see *Vibrational Frequencies*). The default value (100 cm^{-1}) is consistent with the original Quasi-RRHO paper[280], but other papers may choose different values, such as 50 cm^{-1}. Meanwhile, ORCA's Quasi-RRHO implementation deviates from the original paper in the choice of "average molecular moment of inertia" B_{av}; while in the original paper it is chosen as a fixed value $10^{-44}\text{kg}\cdot\text{m}^2$, in ORCA it is given as the isotropically averaged moment of inertia of the actual molecule at hand. This is theoretically more justified than using the same moment of inertia for molecules of different sizes, although the resulting difference in the Gibbs free energies is rather small, usually within 0.1 kcal/mol.

Note that the rotational contribution to the entropy is calculated using the expressions given by Herzberg [563] including the symmetry number obtained from the order of the point group.[1] While this is a good approximation, one might want to modify the symmetry number or use a different expression [564]. For this purpose, the rotational constants (in cm^{-1}) of the molecule are also given in the thermochemistry output.

For example let us calculate a number for the oxygen-oxygen dissociation energy in the H_2O_2 molecule. First run the following jobs:

```
# Calculate a value for the O-O bond strength in H2O2
! B3LYP DEF2-TZVP OPT FREQ BOHRS

* xyz 0 1
 O          -1.396288    -0.075107     0.052125
 O           1.396289    -0.016261    -0.089970
 H          -1.775703     1.309756    -1.111179
 H           1.775687     0.140443     1.711854
*
```

```
# Now the OH radical job
! B3LYP DEF2-TZVP OPT FREQ BOHRS

* xyz 0 2
 O          -1.396288    -0.075107     0.052125
 H          -1.775703     1.309756    -1.111179
*
```

[1] the corresponding equation for the partition function (assuming sufficiently high temperatures) of a linear molecule is $Q_{int} = \frac{kT}{\sigma hcB}$ and for non-linear molecules $Q_{int} = \frac{1}{\sigma}\sqrt{\frac{\pi}{ABC}(\frac{kT}{hc})^3}$. A, B and C are the corresponding rotational constants, σ is the symmetry number. If you want to choose a different symmetry number, ORCA also provides a table with the values for this entropy contribution for other symmetry numbers. Herzberg reports the following symmetry numbers for the point groups C_1,C_i,C_s: 1; C_2,C_{2v}, C_{2h}: 2; C_3,C_{3v},C_{3h}: 3; C_4,C_{4v},C_{4h}: 4;C_6, C_{6v}, C_{6h}: 6; D_2, D_{2d}, D_{2h} =V_h: 4; D_3, D_{3d}, D_{3h}: 6; D_4, D_{4d}, D_{4h}: 8; D_6, D_{6d}, D_{6h}: 12; S_6: 3; $C_{\infty v}$: 1; $D_{\infty h}$: 2;T,T_d: 12; O_h: 24.

The first job gives you the following output following the frequency calculation:

```
-------------------------
THERMOCHEMISTRY AT 298.15K
-------------------------

Temperature         ... 298.15 K
Pressure            ... 1.00 atm
Total Mass          ... 34.01 AMU

Throughout the following assumptions are being made:
  (1) The electronic state is orbitally nondegenerate
  (2) There are no thermally accessible electronically excited states
  (3) Hindered rotations indicated by low frequency modes are not
      treated as such but are treated as vibrations and this may
      cause some error
  (4) All equations used are the standard statistical mechanics
      equations for an ideal gas
  (5) All vibrations are strictly harmonic

freq.     370.67  E(vib)   ...       0.21
freq.     947.27  E(vib)   ...       0.03
freq.    1313.46  E(vib)   ...       0.01
freq.    1440.24  E(vib)   ...       0.00
freq.    3739.49  E(vib)   ...       0.00
freq.    3739.86  E(vib)   ...       0.00

------------
INNER ENERGY
------------

The inner energy is: U= E(el) + E(ZPE) + E(vib) + E(rot) + E(trans)
    E(el)   - is the total energy from the electronic structure calculation
              = E(kin-el) + E(nuc-el) + E(el-el) + E(nuc-nuc)
    E(ZPE)  - the zero temperature vibrational energy from the frequency␣
↪calculation
    E(vib)  - the finite temperature correction to E(ZPE) due to population
              of excited vibrational states
    E(rot)  - is the rotational thermal energy
    E(trans)- is the translational thermal energy

Summary of contributions to the inner energy U:
Electronic energy                ...   -151.55083691 Eh
Zero point energy                ...      0.02631509 Eh      16.51 kcal/mol
Thermal vibrational correction   ...      0.00040105 Eh       0.25 kcal/mol
Thermal rotational correction    ...      0.00141627 Eh       0.89 kcal/mol
Thermal translational correction ...      0.00141627 Eh       0.89 kcal/mol
-----------------------------------------------------------------------
Total thermal energy                   -151.52128823 Eh


Summary of corrections to the electronic energy:
(perhaps to be used in another calculation)
Total thermal correction                 0.00323359 Eh       2.03 kcal/mol
Non-thermal (ZPE) correction             0.02631509 Eh      16.51 kcal/mol
-----------------------------------------------------------------------
Total correction                         0.02954868 Eh      18.54 kcal/mol


--------
ENTHALPY
--------
```

(continues on next page)

(continued from previous page)

```
The enthalpy is H = U + kB*T
                kB is Boltzmann's constant
Total free energy                ...   -151.52129054 Eh
Thermal Enthalpy correction      ...      0.00094421 Eh       0.59 kcal/mol
-------------------------------------------------------------------
Total Enthalpy                   ...   -151.52034633 Eh


Note: Rotational entropy computed according to Herzberg
Infrared and Raman Spectra, Chapter V,1, Van Nostrand Reinhold, 1945
Point Group:  C2, Symmetry Number:   2
Rotational constants in cm-1:    10.087644     0.882994     0.851333

Vibrational entropy computed according to the QRRHO of S. Grimme
Chem.Eur.J. 2012 18 9955


-------
ENTROPY
-------

The entropy contributions are T*S = T*(S(el)+S(vib)+S(rot)+S(trans))
     S(el)   - electronic entropy
     S(vib)  - vibrational entropy
     S(rot)  - rotational entropy
     S(trans)- translational entropy
The entropies will be listed as multiplied by the temperature to get
units of energy

Electronic entropy               ...      0.00000000 Eh      0.00 kcal/mol
Vibrational entropy              ...      0.00059250 Eh      0.37 kcal/mol
Rotational entropy               ...      0.00789993 Eh      4.96 kcal/mol
Translational entropy            ...      0.01734394 Eh     10.88 kcal/mol
-------------------------------------------------------------------
Final entropy term               ...      0.02583637 Eh     16.21 kcal/mol

In case the symmetry of your molecule has not been determined correctly
or in case you have a reason to use a different symmetry number we print
out the resulting rotational entropy values for sn=1,12 :
 -------------------------------------------------------
|  sn= 1 | S(rot)=       0.00855439 Eh      5.37 kcal/mol|
|  sn= 2 | S(rot)=       0.00789993 Eh      4.96 kcal/mol|
|  sn= 3 | S(rot)=       0.00751710 Eh      4.72 kcal/mol|
|  sn= 4 | S(rot)=       0.00724548 Eh      4.55 kcal/mol|
|  sn= 5 | S(rot)=       0.00703479 Eh      4.41 kcal/mol|
|  sn= 6 | S(rot)=       0.00686265 Eh      4.31 kcal/mol|
|  sn= 7 | S(rot)=       0.00671710 Eh      4.22 kcal/mol|
|  sn= 8 | S(rot)=       0.00659102 Eh      4.14 kcal/mol|
|  sn= 9 | S(rot)=       0.00647981 Eh      4.07 kcal/mol|
|  sn=10 | S(rot)=       0.00638033 Eh      4.00 kcal/mol|
|  sn=11 | S(rot)=       0.00629034 Eh      3.95 kcal/mol|
|  sn=12 | S(rot)=       0.00620819 Eh      3.90 kcal/mol|
 -------------------------------------------------------


-------------------
GIBBS FREE ENERGY
-------------------

The Gibbs free energy is G = H - T*S
```

(continues on next page)

(continued from previous page)

```
Total enthalpy                    ...   -151.52034633 Eh
Total entropy correction          ...     -0.02583637 Eh    -16.21 kcal/mol
-----------------------------------------------------------------------
Final Gibbs free energy         ...   -151.54618270 Eh

For completeness - the Gibbs free energy minus the electronic energy
G-E(el)                           ...      0.00465413 Eh      2.92 kcal/mol
```

And similarly for the OH-radical job.

```
------------
INNER ENERGY
------------

The inner energy is: U= E(el) + E(ZPE) + E(vib) + E(rot) + E(trans)
    E(el)   - is the total energy from the electronic structure calculation
              = E(kin-el) + E(nuc-el) + E(el-el) + E(nuc-nuc)
    E(ZPE)  - the zero temperature vibrational energy from the frequency␣
↪calculation
    E(vib)  - the finite temperature correction to E(ZPE) due to population
              of excited vibrational states
    E(rot)  - is the rotational thermal energy
    E(trans)- is the translational thermal energy

Summary of contributions to the inner energy U:
Electronic energy                ...    -75.73492538 Eh
Zero point energy                ...      0.00837287 Eh       5.25 kcal/mol
Thermal vibrational correction   ...      0.00000000 Eh       0.00 kcal/mol
Thermal rotational correction    ...      0.00094418 Eh       0.59 kcal/mol
Thermal translational correction ...      0.00141627 Eh       0.89 kcal/mol
-----------------------------------------------------------------------
Total thermal energy                    -75.72419205 Eh


Summary of corrections to the electronic energy:
(perhaps to be used in another calculation)
Total thermal correction                  0.00236045 Eh       1.48 kcal/mol
Non-thermal (ZPE) correction              0.00837287 Eh       5.25 kcal/mol
-----------------------------------------------------------------------
Total correction                          0.01073332 Eh       6.74 kcal/mol


--------
ENTHALPY
--------

The enthalpy is H = U + kB*T
                kB is Boltzmann's constant
Total free energy                ...    -75.72419205 Eh
Thermal Enthalpy correction      ...      0.00094421 Eh       0.59 kcal/mol
-----------------------------------------------------------------------
Total Enthalpy                   ...    -75.72324785 Eh


Note: Rotational entropy computed according to Herzberg
Infrared and Raman Spectra, Chapter V,1, Van Nostrand Reinhold, 1945
Point Group:  C2v, Symmetry Number:   1
Rotational constants in cm-1:     0.000000    18.628159    18.628159

Vibrational entropy computed according to the QRRHO of S. Grimme
Chem.Eur.J. 2012 18 9955
```

(continues on next page)

(continued from previous page)

```
-------
ENTROPY
-------

The entropy contributions are T*S = T*(S(el)+S(vib)+S(rot)+S(trans))
     S(el)   - electronic entropy
     S(vib)  - vibrational entropy
     S(rot)  - rotational entropy
     S(trans)- translational entropy
The entropies will be listed as multiplied by the temperature to get
units of energy


Note: Rotational entropy computed according to Herzberg
Infrared and Raman Spectra, Chapter V,1, Van Nostrand Reinhold, 1945
Point Group:  C2v, Symmetry Number:   1
Rotational constants in cm-1:     0.000000    18.628159    18.628159

Vibrational entropy computed according to the QRRHO of S. Grimme
Chem.Eur.J. 2012 18 9955


-------
ENTROPY
-------

The entropy contributions are T*S = T*(S(el)+S(vib)+S(rot)+S(trans))
     S(el)   - electronic entropy
     S(vib)  - vibrational entropy
     S(rot)  - rotational entropy
     S(trans)- translational entropy
The entropies will be listed as multiplied by the temperature to get
units of energy

Electronic entropy                ...      0.00065446 Eh      0.41 kcal/mol
Vibrational entropy               ...      0.00000000 Eh      0.00 kcal/mol
Rotational entropy                ...      0.00321884 Eh      2.02 kcal/mol
Translational entropy             ...      0.01636225 Eh     10.27 kcal/mol
-------------------------------------------------------------------------
Final entropy term                ...      0.02023555 Eh     12.70 kcal/mol


-------------------
GIBBS FREE ENERGY
-------------------

The Gibbs free energy is G = H - T*S

Total enthalpy                    ...    -75.72324785 Eh
Total entropy correction          ...     -0.02023555 Eh    -12.70 kcal/mol
-------------------------------------------------------------------------
Final Gibbs free energy         ...    -75.74348340 Eh

For completeness - the Gibbs free energy minus the electronic energy
G-E(el)                           ...     -0.00855802 Eh     -5.37 kcal/mol
```

Let us calculate the free energy change for the reaction: $H_2O_2 \rightarrow 2OH$

The individual energy terms are:

Electronic Energy: -151.46985 a.u. -(-151.55084) a.u. = 0.08099 a.u. (50.82 kcal/mol)

Zero-point Energy: 0.01675 a.u. - 0.02631 a.u. = -0.00956 a.u. (-6.00 kcal/mol)

Thermal Correction(translation/rotation): 0.00472 a.u. - 0.00283 a.u. = 0.00189 a.u. (1.19 kcal/mol)

Thermal Enthalpy Correction: 0.00189 a.u. - 0.00094 a.u. = 0.00095 a.u. (0.60 kcal/mol)

Entropy: -0.04047 a.u. -(-0.02584) a.u. = -0.01463 a.u. (-9.18 kcal/mol)

Final ΔG: 37.43 kcal/mol

Thus, both the zero-point energy and the entropy terms contribute significantly to the total free energy change of the reaction. The entropy term is favoring the reaction due to the emergence of new translational and rotational degrees of freedom. The zero-point correction is also favoring the reaction since the zero-point vibrational energy of the O-O bond is lost. The thermal correction and the enthalpy correction are both small.

Tip

- You can run the thermochemistry calculations at several user defined temperatures and pressure by providing the program with a list of temperatures / pressures:

```
%freq  Temp  290, 295, 300     # in Kelvin
        Pressure  1.0, 2.0, 3.0 # in atm
       end
```

- Once a Hessian is available you can rerun the thermochemistry analysis at several user defined temperatures / pressures by providing the keyword PrintThermoChem and providing the name of the Hessian file:

```
! PrintThermoChem
%geom
  inhessname "FreqJob.hess" # default: job-basename.hess
end
%freq  Temp  290, 295, 300     # in Kelvin
        Pressure  1.0, 2.0, 3.0 # in atm
       end
```

4.8.1 Diagonal Born-Oppenheimer Correction

For high-accuracy thermochemistry, one of the higher-order corrections to the energy that is required is the "diagonal Born-Oppenheimer correction" (DBOC). The expression for the DBOC correction is derived by first-order perturbation theory and reads:

$$E_{DBOC} = \left\langle \Psi_e\left(\mathbf{r},\mathbf{R}\right) | \widehat{T}_N | \Psi_e\left(\mathbf{r},\mathbf{R}\right) \right\rangle$$

In order to evaluate the DBOC correction it is required to calculate the response of the orbitals (and the density) with respect to nuclear perturbations, which is the most expensive step of an analytic Hessian calculation. The correction is therefore best be evaluated in parallel with the usual thermochemical corrections. The implementation in ORCA covers the SCF level (e.g. closed-shell and spin-unrestricted Hartree-Fock and DFT).

```
! DBOC
# or NoDBOC to turn it off (which is the default)
```

Relevant Papers:

1. Handy, Nicholas C.; Yamaguchi, Yukio; Schaefer, Henry F., III. The diagonal correction to the Born–Oppenheimer approximation: Its effect on the singlet–triplet splitting of CH2 and other molecular effects. *The Journal of Chemical Physics*, **1986**, 84 (8), 4481–4484. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/84/8/4481/18958051/4481_1_online.pdf, DOI: 10.1063/1.450020.

4.8.2 Spin-Orbit Coupling Correction to the Total Energy

Another correction that is required for high-accuracy thermochemistry is the estimation of the energetic effect of Spin-Orbit-Coupling (SOC) on the total energy. While most of ORCA treats the SOC as a perturbation and estimates its effect on molecular properties and spectra, the total energy contribution would require relativistic two-component theory. This is not yet implemented in ORCA. Thus, for the time being, there is a first-order perturbation correction for the SOC that proceeds as follows:

The SCF energy is:

$$E_{SCF} = E_{Nuc} + \frac{1}{2} Tr\left(\mathbf{P}, \mathbf{H} + \mathbf{F}\right)$$

In two-component theory, the Fock operator is:

$$\overline{\mathbf{F}} = \begin{pmatrix} \mathbf{F}^{\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^{\beta} \end{pmatrix} + \begin{pmatrix} \frac{1}{2}\mathbf{T}^{z} & \frac{1}{2}\mathbf{T}^{x} + \frac{i}{2}\mathbf{T}^{y} \\ \frac{1}{2}\mathbf{T}^{x} - \frac{i}{2}\mathbf{T}^{y} & -\frac{1}{2}\mathbf{T}^{z} \end{pmatrix}$$

Where $T^{x,y,z}$ are the 3 components of the SOC matrix.

We then solve the eigenvalue problem:

$$\overline{\mathbf{F}}\overline{\mathbf{C}} = \overline{\varepsilon}\overline{\mathbf{S}}\overline{\mathbf{C}}$$

With the two-component overlap matrix

$$\overline{\mathbf{S}} = \begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}$$

Now we calculate the 2C density matrix from the eigenvectors

$$\overline{\mathbf{P}} = \breve{\mathbf{C}}^{*}\breve{\mathbf{C}}^{T}$$

Where $\breve{C}$ is the occupied part of $\overline{C}$. This then leads to the approximate 2 component energy:

$$E_{2C} = E_{Nuc} + \frac{1}{2} Tr\left(\overline{\mathbf{P}}, \overline{\mathbf{H}} + \overline{\mathbf{F}}\right)$$

And the SOC correction is then simply:

$$\Delta_{SOC} = E_{2C} - E_{SCF}$$

$$= \frac{1}{2} Tr\left(\overline{\mathbf{P}}, \overline{\mathbf{H}} + \overline{\mathbf{F}}\right) - \frac{1}{2} Tr\left(\mathbf{P}, \mathbf{H} + \mathbf{F}\right)$$

This is obviously neglecting the relaxation of the orbitals due to the SOC. The approximation is necessarily rather crude and consequently, this term does require a bit of empirical scaling in actual applications to reach high accuracy.

Relevant Papers:

- Kayal, R.; Neese, F. **2025**, in preparation

4.8.3 High Accuracy Thermochemistry: the HEAT protocol

The HEAT (High accuracy Extrapolated Ab initio Thermochemistry) protocol was established by Gauss, Stanton, Kallay and co-workers and aims at high accuracy total energies that are systematically within 1 kJ/mol of the most accurate experimental energies.

The original HEAT formula contains eight terms:

$$E_{HEAT} = E_{HF}^{\infty} + \Delta E_{CCSD(T)}^{\infty} + \Delta E_{CCSDT} + \Delta E_{CCSDTQ} + \Delta E_{rel} + \Delta E_{ZPE} + \Delta E_{DBOC} + \Delta E_{SOC}$$

Where the individual contributions are:

Term	Original Definition	Implementation in ORCA
Geometry	Optimized using CCSD(T)/cc-pVQZ	Same
E_{HF}^{∞}	Complete basis set limit extrapolated Hartree-Fock energy using aug-cc-pCVXZ (X=T,Q,5) and the formula $E_{HF}^{X} = E_{HF}^{\infty} + a\exp(-bX)$ E_{HF}^{∞}, a, b are determined from the three energies	Same
$\Delta E_{CCSD(T)}^{\infty}$	Complete basis set limit extrapolated CCSD(T) energy using aug-cc-pCVXZ (X=Q,5) and the formula $E_{CCSD(T)}^{X} = E_{CCSD(T)}^{\infty} + aX^{-3}$	Same
ΔE_{CCSDT}	Compensation of the difference between CCSD(T) and CCSDT. Difference between two-point extrapolated frozen-core CCSDT and CCSD(T) using $\Delta E_{CCSDT} = E_{CCSDT}^{TQ}(FC) - E_{CCSD(T)}^{TQ}(FC)$	Same
ΔE_{CCSDTQ}	Compensation of the difference between CCSDT and CCSDTQ using the cc-pVDZ basis set $\Delta E_{CCSDTQ} = E_{CCSDTQ}^{DZ}(FC) - E_{CCSDT}^{DZ}(FC)$	Same
ΔE_{rel}	Effects of scalar relativity. Expectation value of the CCSD(T)/ aug-cc-pCVTZ-DK density with the Darwin and Mass-Velocity Hamiltonians.	Estimated using the difference in CCSD(T) energies betweeen aug-cc-pCVTZ and aug-cc-pCVTZ-DK basis sets (with the DKH Hamiltonian).
ΔE_{ZPE}	Zero-point energy using an anharmonic Force Field calculated at CCSD(T)/cc-pVQZ $\Delta E_{ZPE} = \frac{1}{2}\sum_i \omega_i + \frac{1}{4}\sum_{i \leq j} x_{ij}$ Where ω and x are harmonic and cubic force constants.	With the harmonic approximation at the CCSD(T)/cc-pVQZ level with a scaling factor for linear and symmetric top molecules, with the VPT2 module for rest of the molecules.
ΔE_{DBOC}	Diagonal Born-Oppenheimer correction calculated at the SCF level.	Same
ΔE_{SOC}	Spin-Orbit-Coupling correction. Full Valence CASSCF with singles- and doubles CI together with spin-orbit effective core potentials.	SCF level first order correction as described in *Spin-Orbit Coupling Correction to the Total Energy*

continues on next page

Table 4.7 – continued from previous page

Term	Original Definition	Implementation in ORCA
		and empirically scaled.

Thus, for the most part, the ORCA implementation follows the original prescription with some limited changes that arise from the unavailability of all energy components in ORCA. The HEAT protocol is computed by a compound script HEAT.cmp that is delivered with ORCA.

Relevant Papers:

1. Tajti, Attila; Szalay, Péter G.; Császár, Attila G.; Kállay, Mihály; Gauss, Jürgen; Valeev, Edward F.; Flowers, Bradley A.; Vázquez, Juana; Stanton, John F. HEAT: High accuracy extrapolated ab initio thermochemistry. *The Journal of Chemical Physics*, **2004**, 121 (23), 11599–11613. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/121/23/11599/19246319/11599_1_online.pdf, DOI: 10.1063/1.1811608.

- Kayal, R.; Neese, F. **2025**, in preparation

4.9 Conical Intersections

The minima in the conical intersection seam-space between two states (named here I and J) can be found by using regular geometry optimization algorithms, except that the gradient to be optimized is [565]:

$$\mathbf{g} = \mathbf{g'}_{diff} + \mathbf{P}\mathbf{g}_{mean}$$

where $\mathbf{g'}_{diff} = 2(E_I - E_J)(\partial E_I/\partial q - \partial E_J/\partial q)$ is parallel to the gradient difference vector; $\mathbf{g}_{mean}$ is the gradient mean and $\mathbf{P}$ is a projection matrix that projects out the gradient difference ($\mathbf{x}$) and non-adiabatic coupling ($\mathbf{y}$) direction components:

$$\mathbf{P} = \mathbf{1} - \mathbf{x}\mathbf{x}^{\mathbf{T}} - \mathbf{y}\mathbf{y}^{\mathbf{T}}$$

Now we have three approaches to solve this problem in ORCA, that will be explained next.

4.9.1 Gradient Projection

This is exactly what has been described above, and will be chosen as default whenever NACMEs between I and J are available. It is in principle the fastest and most accurate method. It can be invoked by setting:

```
%CONICAL
  METHOD GRADIENT_PROJECTION  #or simply GP
END
```

OBS.: Turning on the ETF (see Sec. Section 5.6.18) can improve the optimization when using the full Gradient Projection method.

4.9.2 Gradient Projection without NACME

It is an approximation to the method above, that one gets by completely neglecting the NACMEs. It is essentially equivalent to finding a surface crossing point, and will not necessarily find minima inside the CI seam-space, although the final ΔE_{IJ} should be zero.

```
%CONICAL
  METHOD GP_NONACME
END
```

4.9.3 Updated Branching Plane

Here the idea is to start from a guess NACME, which is any unit vector perpendicular to **x**, and do a progressive update on it, similar to the BFGS update on the Hessian [565]. The "Branching Plane" defined by **x** and **y** then becomes iteratively more accurate until covergence is achieved. It has been shown to be quite accurate and is the default whenever NACMEs are not available. Can be used with:

```
%CONICAL
  METHOD UBP
END
```

Finally, the ΔE_{IJ} energy threshold for the optimization can be altered with:

```
%CONICAL
  ETOL 1e-4 #default
END
```

As a further example for such an analysis we show the conical intersection optimization of the ground and first excited state of singlet ethylene.

> **Note**
>
> This is currently only available using TD-DFT, will be expanded in future versions. More details about the specific options on *Excited States via RPA, CIS, TD-DFT and SF-TDA*.

> **Note**
>
> Even though locating the CI of a TD-DFT excited state and the reference state is supported, it is not the recommended way of finding the ground state-excited state CI, because such CIs are not described properly by TD-DFT (in particular, TD-DFT even predicts the wrong dimensionality for the intersection space). Instead, it is advised to use SF-TD-DFT for this purpose, e.g. use the T_1 state as the reference state, and calculate both the S_0 and S_1 states as excited states. (*vide infra*)

```
!B3LYP DEF2-SVP CI-OPT
%TDDFT IROOT 1 END
* xyz 0 1
C         0.595560237     -0.010483480     -0.000284187
C        -0.831313750      0.167231832      0.001482505
H        -1.381857976      0.227877089      0.963419721
H         1.265119434      0.874806815      0.006897459
H        -1.382258208      0.243775568     -0.959090898
H         1.027489724     -1.032962768     -0.008829646
*
```

> **Tip**
>
> You can often use a structure between the optimized structures of both states for your CI-optimization. If this doesn't work, you might try a scan to get a better initial guess.

The results of the CI-optimization are given in the following output. The energy difference between the ground and excited state is printed as E diff. (CI), being reasonably close for a conical intersection. For a description of the calculation of the non-adiabatic couplings at this geometry, see section Section 5.6.18.

```
                                    .--------------------.
                     ----------------------|Geometry convergence|------------------------
```

(continues on next page)

(continued from previous page)

```
              Item                 value                   Tolerance       Converged
              ---------------------------------------------------------------------
              Energy change        0.0000164283            0.0000050000      NO
              E diff. (CI)         0.0000025162            0.0001000000      YES
              RMS gradient         0.0000068173            0.0001000000      YES
              MAX gradient         0.0000136891            0.0003000000      YES
              RMS step             0.0000358228            0.0020000000      YES
              MAX step             0.0000821130            0.0040000000      YES
              .......................................................
              Max(Bonds)      0.0000      Max(Angles)    0.00
              Max(Dihed)        0.00      Max(Improp)    0.00
              ---------------------------------------------------------------------

              Everything but the energy has converged. However, the energy
              appears to be close enough to convergence to make sure that the
              final evaluation at the new geometry represents the equilibrium␣
↪energy.
              Convergence will therefore be signaled now


                        ************************HURRAY********************
                        ***        THE OPTIMIZATION HAS CONVERGED     ***
                        **************************************************


    ---------------------------------------------------------------------------
                          Redundant Internal Coordinates

                           --- Optimized Parameters ---
                             (Angstroem and degrees)

        Definition                    OldVal   dE/dq     Step     FinalVal
    ---------------------------------------------------------------------------
     1. B(C   1,C   0)                1.3254 -0.000005 -0.0000    1.3254
     2. B(H   2,C   1)                1.1270  0.000004 -0.0000    1.1270
     3. B(H   3,C   0)                1.1271 -0.000002  0.0000    1.1271
     4. B(H   4,C   1)                1.1271  0.000000 -0.0000    1.1271
     5. B(H   5,C   0)                1.1271 -0.000002  0.0000    1.1271
     6. A(H   3,C   0,H   5)          106.00  0.000001   -0.00    106.00
     7. A(C   1,C   0,H   5)          126.97 -0.000013    0.00    126.97
     8. A(C   1,C   0,H   3)          127.03  0.000011   -0.00    127.03
     9. A(C   0,C   1,H   4)          127.03  0.000013   -0.00    127.03
    10. A(H   2,C   1,H   4)          106.01 -0.000000    0.00    106.01
    11. A(C   0,C   1,H   2)          127.04  0.000009   -0.00    127.04
    12. D(H   2,C   1,C   0,H   5)    106.86 -0.000005    0.00    106.86
    13. D(H   4,C   1,C   0,H   3)    106.76 -0.000005    0.00    106.76
    14. D(H   4,C   1,C   0,H   5)    -73.62 -0.000004    0.00    -73.61
    15. D(H   2,C   1,C   0,H   3)    -72.76 -0.000005    0.01    -72.76

    ---------------------------------------------------------------------------
                 *******************************************************
                 *** FINAL ENERGY EVALUATION AT THE STATIONARY POINT ***
                 ***                (AFTER   12 CYCLES)                ***
                 *******************************************************
```

4.9.4 CI Minima Between Excited States

In an analogous way, the conical intersection minima between two excited states can be requested by selection both an IROOT and a JROOT, shown below.

```
!B3LYP DEF2-SVP CI-OPT
%TDDFT IROOT 2
       JROOT 1
       #IROOTMULT TRIPLET would search in the triplet PES
       #SF TRUE would search for the S0-S1 CI from a T1 reference, using SF-TD-DFT
       # (but remember to set the multiplicity as 3 instead of 1)
END
* xyz 0 1
C        0.595560237     -0.010483480     -0.000284187
C       -0.831313750      0.167231832      0.001482505
H       -1.381857976      0.227877089      0.963419721
H        1.265119434      0.874806815      0.006897459
H       -1.382258208      0.243775568     -0.959090898
H        1.027489724     -1.032962768     -0.008829646
*
```

4.10 Minimum Energy Crossing Points (MECP)

The MECP optimization allows the user to optimize to structures where two different potential energy surfaces (PES1 and PES2) cross each other. In this optimization two conditions apply: the energy E_1 of PES1 is minimized while at the same time the energy difference $(E_1 - E_2)^2$ of both surfaces is minimized. For the implementation we follow in principle the suggestions of Harvey et al. in [541].

For the minimization two different gradients are constructed:

The first gradient chosen for the minimization is

$$\mathbf{f} = \frac{\partial}{\partial q}(E_1 - E_2)^2 = 2(E_1 - E_2) \cdot x_1 \tag{4.25}$$

where x_1 is the gradient difference vector

$$x_1 = \left[\left(\frac{\partial E_1}{\partial q}\right) - \left(\frac{\partial E_2}{\partial q}\right)\right] \tag{4.26}$$

which is orthogonal to the crossing hyperline near the MECP.

The gradient

$$\mathbf{g} = \left(\frac{\partial E_1}{\partial q}\right) - \frac{x_1}{|x_1|}\left[\left(\frac{\partial E_1}{\partial q}\right) \cdot \frac{x_1}{|x_1|}\right] \tag{4.27}$$

is orthogonal to x_1.

Both gradients are combined to yield the effective surface crossing gradient

$$\mathbf{g}_{\mathrm{SC}} = \mathbf{g} + \mathbf{f} \tag{4.28}$$

The crossing hyperline is defined as the $3N - 7$ dimensional subspace of PES1, which is orthogonal to x_1. In the MECP optimization we want to find the point of lowest energy within this subspace.

Our calculation of normal modes and force constants for movements along the crossing hyperline differ from the one proposed by Harvey et al. A standard frequency analysis cannot be performed, but a similar procedure is applied:

Let us regard the second-order Taylor expansion for the energy of both surfaces near the MECP for a displacement along the crossing hyperline (orthogonal to x_1):

$$\mathbf{E}_A = E_{\mathrm{MECP}} + \frac{1}{2}\Delta q^T H_{\mathrm{eff},A} \Delta q \tag{4.29}$$

with:

$\mathbf{E}_A$	Energy E_1 on PES1 or E_2 on PES2
$H_{\text{eff},A}$	effective Hessian for PES1 or PES2
Δq	displacement along the crossing hyperline

Diagonalization of this effective Hessian gives us the normal modes of the crossing hyperline and thus allows us to decide whether the MECP optimization converged to a minimum in the $3N-7$ dimensional subspace of the crossing hyperline.

The procedure for the calculation of the effective Hessian is now as follows: For each of both surfaces the second derivative matrix is calculated. Then the 6 rotations and translations and additionally the direction of the gradient difference vector x_1 (this ensures that movement orthogonal to the crossing hyperline, for which we do NOT satisfy the conditions of a stationary point, is excluded) are projected out from the Hessian matrix.

As an example we show the MECP optimization of the quartet and sextet state of $[FeO]^+$.

```
! B3LYP TZVP Opt SurfCrossOpt SlowConv

%mecp
   Mult 4
end

* xyz +1 6
Fe 0.000000     0.000000     0.000000
O  0.000000     0.000000     1.670000
*
```

Tip

You can often use a minimum or TS structure of one of the two spin states as initial guess for your MECP-optimization. If this doesn't work, you might try a scan to get a better initial guess.

The results of the MECP optimization are given in the following output. The distance where both surfaces cross is at 1.994 Å. In this simple example there is only one degree of freedom and we can also locate the MECP via a parameter scan. The results of the scan are given in Fig. 4.5 for comparison. Here we see that the crossing occurs at a Fe-O-distance of around 2 Å.

For systems with more than two atoms a scan is not sufficient any more and you have to use the MECP optimization.

```
************************HURRAY********************
***        THE OPTIMIZATION HAS CONVERGED     ***
*******************************************************

---------------------------------------------------------------------
                   Redundant Internal Coordinates
                   --- Optimized Parameters ---
                   (Angstroem and degrees)
  Definition                    OldVal   dE/dq     Step     FinalVal
   ----------------------------------------------------------------------
1. B(O   1,Fe  0)               1.9940 -0.000001  0.0000    1.9940
   ----------------------------------------------------------------------

      *******************************************************
      *** FINAL ENERGY EVALUATION AT THE STATIONARY POINT ***
      ***                (AFTER    8 CYCLES)               ***
      *******************************************************
```

(continues on next page)

(continued from previous page)

```
   -------------------------------------   ----------------
   Energy difference between both states       -0.000000004
   -------------------------------------   ----------------
```

Fig. 4.5: Parameter scan for the quartet and sextet state of $[FeO]^+$ (B3LYP/SV(P)).

A more realistic example with more than one degree of freedom is the MECP optimization of a structure along the reaction path of the $CH_3O \leftrightarrow CH_2OH$ isomerization.

```
! B3LYP SV SurfCrossOpt SurfCrossNumFreq

%mecp
   Mult 1
end

*xyz 1 3
 C     0.000000     0.000000     0.000000
 H     0.000000     0.000000     1.300000
 H     1.026719     0.000000    -0.363000
 O    -0.879955     0.000000    -1.088889
 H    -0.119662    -0.866667     0.961546
*
```

Note

- To verify that a stationary point in a MECP optimization is a minimum, you have to use an adapted frequency analysis, called by `SurfCrossNumFreq`.

4.10.1 Keywords

For MECP optimizations the following options exist:

```
%mecp
   SurfCrossOpt true  # switches on the MECP optimization
                      #  alternatively use: ! SurfCrossOpt
   SurfCrossNumFreq true # switches on the MECP effective Hessian
                         #  calculation
                         #  alternatively use: ! SurfCrossNumFreq
 # separate MO input for the second spin state (PES2)
   moinp "Myfile.gbw"# MO input for PES2
 # information on the electronic structure of PES 2
   Mult 3            # multiplicity of PES2
   brokenSym 1,1     # broken symmetry for PES2
 # CASSCF options for PES2 (also see the CASSCF chapter)
   casscf_nel 6       # number of active space electrons
   casscf_norb 6      # number of active orbitals
   casscf_mult 1,3    # multiplicities singlet and triplet
   casscf_nroots 4,2  # four singlets, two triplets
   casscf_bweight 2,1 # singlets and triplets weighted 2:1
   casscf_weights[0] = 0.5,0.2,0.2,0.2 # singlet weights
   casscf_weights[1] = 0.7,0.3         # triplet weights
end
```

4.11 GOAT: global geometry optimization and ensemble generator

If instead of trying to optimize a single structure, starting from a given guess geometry, you want to find the **global** minimum or the ensemble around it, ORCA features a Global Optimizer Algorithm (GOAT)[566] inspired by Wales and Doye's basin-hopping [567], Goedecker's minima hopping [568], Simulated Annealing and Taboo Search.

The idea is to start from somewhere on the potential energy surface (PES; red ball on Fig. 4.6), go first to the nearest local minimum (blue ball), and from there start pushing "uphill" on a random direction until a barrier is crossed. Then a new minimum is found and the process is restarted, with another uphill push followed by an optimization. After several of these GOAT iterations (uphill + downhill), if no new global minimum was found between the two last global iterations.

Fig. 4.6: A simple depiction of the difference between a regular geometry optimization (above), and the GOAT global optimizer (below). By using the latter, one finds not only one local minimum but the global one and the conformational ensemble around it.

Since structures are collected along the way to the global minimum, we have in the end not only the global minimum, but also the conformational ensemble for that molecule, meaning all the conformations it **can** have and their relative energies. This is also useful later to compute Boltzmann-averaged spectra and properties.

The idea is similar to what is done with CREST from the group of Prof. Grimme [569], except that no metadynamics is required and thus much less gradient runs are needed. It is thus suitable not only for super fast methods such as XTB and force-fields, but can also be used directly using DFT or with any method available in ORCA.

Please note that there is no *ab initio* way to find global minima for arbitrary unknown functions, and stochastic methods are the most efficient on finding these. The drawback is that it is based on random choices, so that many geometry optimizations are needed - here in the order of $100\times$ the number of atoms. Good news is: these can be efficiently parallelized (even multinode) and this number can be brought down to less than $3\times$ the number of atoms (see below)!

4.11.1 GOAT simple usage example - Histidine

Let's start with a simple example, the amino acid histidine:

Fig. 4.7: A histidine molecule.

By simply looking at its Lewis structure, it is not at all evident that there are actually at least 20 conformers in the 3 kcal/mol range from the global minimum on the XTB PES! In order to find them, one can run:

```
!XTB GOAT #XTB version 6.4.0

* xyz 0 1
N          -0.13033       -0.28496       -0.67901
C           1.30551       -0.36383       -0.41824
C           1.51611       -1.04435        0.94169
O           0.58926       -1.43597        1.64771
C           1.97932        1.02323       -0.44331
H           2.41593        1.27074        0.53237
H           1.24598        1.81337       -0.65132
C           3.04894        1.09545       -1.48009
N           2.77779        1.01389       -2.82650
C           3.97051        1.07618       -3.48654
H           4.04071        1.01588       -4.56429
N           4.97724        1.21154       -2.65295
C           4.41545        1.24248       -1.40412
H           5.02774        1.35929       -0.51882
H           1.85722        0.88822       -3.24287
H          -0.75420       -0.54264        0.08211
H           1.72153       -1.03742       -1.17913
H          -0.47113        0.20468       -1.50142
O           2.79298       -1.20515        1.35271
H           3.59528       -0.86615        0.74156
*
```

The command to call the global optimizer is simply `!GOAT`, like you would with `!OPT`, and its options can be given under the `%GOAT` block as usual. You can give it together with any other method available in ORCA, but it needs to be a fast one because a lot of geometry optimizations need to be done. Here we will just use `GFN2` (or `!XTB`). What will happen next is:

1. First a regular geometry optimization will be done to find the minimum closest to the input structure.
2. With that information in hand, the number of necessary GOAT iterations will be computed and divided among `NWorkers` (8 by default).
3. Each `Worker` has its own parameters and will run a certain number of geometry optimizations.
4. After all workers in a global cycle are done, data will be collected and a new cycle will begin. There will be at least a "Minimum global steps" number of global cycles like this.
5. Once the difference between two global steps is negligible, it stops, collects everything and prints the ensemble energies and a file with all structures.

4.11.2 Understanding the output

After the usual geometry optimization, the output looks like:

```
Global parameters
-----------------
GOAT version                                ... default
Minimum global steps                        ... 3
Number of base workers                      ... 4
Split workers by                            ... 2
Final number of workers                     ... 8
Number of available CPUs                    ... 16
Parameter list (worker : temperature)       ...  0 : 2903.97,  1 : 1451.98,
                                                 2 :  725.99,  3 :  363.00,
                                                 4 : 2903.97,  5 : 1451.98,
                                                 6 :  725.99,  7 :  363.00
GradComp (mean : sigma)                     ... 1.00 : 0.50
Number of atoms                             ... 20
Number of fragments                         ... 1
Flexibility parameter                       ... 0.45
Optimizations per global step               ... 160
Optimizations per worker                    ... 20


Filtering criteria
------------------
RMSD                                        ... 0.125 Angs (atom. pos.)
EnDiff                                      ... 0.100 kcal/mol
RotConst                                    ... 1.00-2.50 %
Maximum Conf. Energy                        ... 6.000 kcal/mol

Thermodynamics
--------------
Ensemble temperature                        ... 298.15 K
Degeneracy of conformers                    ... 1
*No rotamers will be included in Gconf
```

On the top there is some general information. Most important here is that we have 8 workers and 1 CPU, meaning each worker will run only after the other is done. If you want to speed up, just add more CPUs via e.g., `!PAL8` and workers will run in parallel making it 8x faster. GOAT can also run multinode, so feel free to use any number of processors via `%PAL`. In the end the filtering criteria used to differentiate conformers and rotamers is printed.

The default filtering is precisely the same as that of CREST: RMSD of atomic positions together with the rotational constant, considering its anisotropy. For *GOAT-EXPLORE*, the default RMSD metric is based on the eigenvalues of the distance matrix instead, since it is invariant to translations, rotations and the atom ordering, which changes quite often in these cases.

After that, the algorithm starts:

```
-------------------------------------------------------------------------------
Iter  MinTemp     MaxEn       GradComp    NOpt   NProcs    Output
-------------------------------------------------------------------------------
0     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.0.out
1     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.1.out
2      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.2.out
3      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.3.out
4     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.4.out
5     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.5.out
6      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.6.out
7      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.7.out

                              GOAT Global Iter 1
                 Iter    Min En       Sconf       Gconf
```

(continues on next page)

(continued from previous page)

```
                        Hartree      cal/(molK)  kcal/mol
                 ========================================

                    1    -34.346656   4.432       -0.551

0     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.0.out
1     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.1.out
2      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.2.out
3      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.3.out
4     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.4.out
5     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.5.out
6      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.6.out
7      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.7.out

                           GOAT Global Iter 2
                 Iter    Min En       Sconf       Gconf
                         Hartree      cal/(molK)  kcal/mol
                 ========================================

                    1    -34.346656   4.432       -0.551
                    2    -34.346656   4.528       -0.559

0     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.0.out
1     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.1.out
2      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.2.out
3      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.3.out
4     2903.97     60.00    1.00 : 0.50      20        2    HIS.goat.0.4.out
5     1451.98     60.00    1.00 : 0.50      20        2    HIS.goat.0.5.out
6      725.99     60.00    1.00 : 0.50      20        2    HIS.goat.0.6.out
7      363.00     60.00    1.00 : 0.50      20        2    HIS.goat.0.7.out

                           GOAT Global Iter 3
                 Iter    Min En       Sconf       Gconf
                         Hartree      cal/(molK)  kcal/mol
                 ========================================

                    1    -34.346656   4.432       -0.551
                    2    -34.346656   4.528       -0.559
                    3    -34.346656   4.541       -0.560

                                     Global minimum found!
                         Writing structure to HIS.globalminimum.xyz
```

On the top header one can see what are the temperatures used, the maximum energy allowed during an uphill step, the maximum coefficient for gradient reflection, the number of optimizations done per worker, the number of processors used for each and the local output file name.

The names of the output files are chosen as `BaseName.goat.globaliteration.workernumber.out`. These are deleted after the run by default, but can be kept by setting `KEEPWORKERDATA TRUE` under `%GOAT`.

During each global iteration, the minimum energy so far, the conformational entropy S_{conf} and the conformational Gibbs free energy G_{conf} are printed. Since **there are no rotamers here**, the entropy is calculated only on the basis of the conformer energies and its convergence hints to the completeness of the ensemble created.

4.11.3 The final ensemble

In this case, as you can see, it already found the global minimum after the first global cycle with E = -34.346656 Hartree, but it keeps running for at least 3 cycles, following the defaults. This happens because it is a small molecule, but it is not necessarily so and more cycles will be done if needed.

The final relative energies of the ensemble are printed afterwards, together with a `BaseName.finalensemble.xyz` file:

```
                 # Final ensemble info #
                 Conformer     Energy     Degen.   % total   % cumul.
                               (kcal/mol)
                 -------------------------------------------------------
                         0     0.000          1      37.96      37.96
                         1     0.503          1      16.25      54.20
                         2     0.914          1       8.11      62.32
                         3     1.159          1       5.37      67.68
                         4     1.297          1       4.25      71.94
                         5     1.320          1       4.09      76.03
                                            (...)
                        41     5.180          1       0.01      99.98
                        42     5.199          1       0.01      99.99
                        43     5.491          1       0.00      99.99
                        44     5.547          1       0.00      99.99
                        45     5.861          1       0.00     100.00

Conformers below 3 kcal/mol: 22
Lowest energy conformer    : -34.346656 Eh
Sconf at 298.15 K          :  4.54 cal/(molK)
Gconf at 298.15 K          : -0.56 kcal/mol

Writing final ensemble to HIS.finalensemble.xyz
```

Just for the record, here is how the four lowest lying conformers look like:

Fig. 4.8: The four lowest conformers found for histidine on the XTB PES.

Whenever using `GOAT`, the `EnforceStrictConvergence` policy from ORCA's optimizer is set to `TRUE`. This is recommended here the ensure equal criteria for different molecules in the ensemble. It can be turned off by setting `%GEOM EnforceStrictConvergence FALSE END`.

The regular optimization thresholds are already good, but it might also be a good idea to use `!TIGHTOPT` to make sure all your ensemble molecules are well converged, specially for the `GOAT-EXPLORE`!

4.11.4 GOAT-ENTROPY: expanding ensemble completeness by maximizing entropy

If you want to be as complete as possible in terms of the ensemble, you can use the `!GOAT-ENTROPY` keyword instead. This will not only try to find the global minimum until the energy is converged, but will actually only stop when the ΔS_{conf} also converges to less than 0.1 cal/(molK), which is equivalent to maximizing the conformational entropy (the threshold can be altered – see keyword list below).

This will push the algorithm so that all conformers around the global minimum should be found together with it. Both temperature and ΔS_{conf} can be changed via specific keywords shown at the list below. A higher temperature will make the ΔS_{conf} more sensitive to changes in high energy conformational regions and should make the search even more complete.

Being more explicit, the conformational entropy, enthalpy and Gibbs free energy are calculated according to:

$$S_{\text{conf}} = R\left[\ln \sum g_i' e^{-E_i\beta} + \frac{\sum g_i'(E_i\beta)e^{-E_i\beta}}{\sum g_i' e^{-E_i\beta}}\right]$$

$$\left[H(T) - H(0)\right]_{\text{conf}} = RT\frac{\sum g_i(E_i\beta)e^{-E_i\beta}}{\sum g_i e^{-E_i\beta}}$$

where $\beta = \frac{1}{k_B T}$ and g_i is the "degeneracy" of conformer i, i.e. its number of rotamers. This is the correct approach to deal with degenerate states [570]. The only difference from the reference above is that g_i' is always one and we don't discriminate any factor for "geometrical enantiomers".

4.11.5 More on the ΔS_{conf}

It is important to say that, by default, the ΔS_{conf} **is not the same as that found by a default CREST run**. There it includes also the rotamer degeneracy on the calculation of the entropy, while here that is 1. The reasons for that are:

1. There are formal arguments for using only one, assuming that rotamers are indistinguishable. Please check Grimme's reference [570] for details.
2. For systems with many rotamers, e.g. molecules with 3 *tert*-butyl groups which give rise to at least (27^3) 19683 rotamers per conformer, the algorithm will never find all of these anyway.
3. When calling `GOAT-ENTROPY`, it is the entropy of the **conformers** that will be maximized, not of the ensemble. That guarantees the maximal distribution of different conformers during these searches.

Once the final ensemble is found and if you know how many rotamers per conformer you have (assuming a constant number, like the *tert*-butyl case), one can reset that number by using `READENSEMBLE "ensemble.xyz"` to read it and `CONFDEGEN` to set a degeneracy. In the previous example that would be `CONFDEGEN 19683` and would give you the desired ΔG_{conf} for that given ensemble.

Finding rotamers automatically

It is always possible to switch on the automatic search for rotamers and their degeneracy by setting `CONFDEGEN AUTO` if that is what you want. They will be added to the ensemble instead of being filtered out and the full ensemble will be saved in files named `.confrot.xyz`.

Another approach would be to assume that, since the algorithm might find all rotamers for some conformers but not for all, one might set the degeneracy of all equal to the maximum value found so far (`CONFDEGEN AUTOMAX`). Let's take as an example a system with a *tert*-butyl + a methyl group with 81 rotamers per conformer. GOAT will hardly find 81 rotamers for every single conformer, but if it finds them for one conformer, all the others will also have that same number.

Please be aware that there are cases where different conformers might have different numbers of rotamers (e.g. decane or long alkyl chains), and these cases should be treated with care.

4.11.6 GOAT-EXPLORE: global minima of atomic clusters or topology-free PES searches

In case you want to find the lowest energy conformer for a cluster or don't want to keep the initial topology at all, you can use the `!GOAT-EXPLORE` option instead. This will possibly break all bonds and find the lowest energy structure for that given set of atoms, be that a nanoparticle or an organic molecule.

For instance, let's find the minimum of an Au_8 nanoparticle on the GFN1 PES, starting from just a random agglomeration of gold atoms:

```
!XTB1 GOAT-EXPLORE PAL16 #XTB version 6.4.0

%GOAT NWORKERS 16 END

* xyz 0 1
Au        -1.39858        2.62611       -0.79278
Au        -2.50552       -0.07122        0.67538
Au        -0.52174       -2.57892       -0.02415
Au         0.78881        0.39733        0.21816
Au         1.21116        1.90621       -2.64617
Au        -1.19205       -0.30099       -2.30429
Au        -0.33808       -1.07129        2.90822
Au        -0.85565        2.15342        2.41956
*
```

Please note that there is a minimum number of optimizations per worker that must be respected in order for the algorithm to make sense. Otherwise, on the limit, one optimization per worker would mean almost nothing happens. This minimum number is `max(N, 15)` for the regular GOAT and `max(3N, 45)` for *`GOAT-EXPLORE`* and *`GOAT-REACT`* (see below), where N is the number of atoms. The searches using free topology are more demanding because there are many more degrees of freedom.

When the minimum number of optimizations per worker is reached, the information is printed on the bottom of the header as:

```
GOAT version                                ... explore
Minimum global steps                        ... 3
Number of base workers                      ... 4
Split workers by                            ... 4
Final number of workers                     ... 16
Number of available CPUs                    ... 1
Parameter list (worker : temperature)       ...  0 : 2903.97,  1 : 1451.98,
                                                 2 :  725.99,  3 :  363.00,
                                                 4 : 2903.97,  5 : 1451.98,
                                                 6 :  725.99,  7 :  363.00,
                                                 8 : 2903.97,  9 : 1451.98,
                                                 10 :  725.99, 11 :  363.00,
```

(continues on next page)

(continued from previous page)

```
                                                 12 : 2903.97, 13 : 1451.98,
                                                 14 :  725.99, 15 :  363.00
GradComp (mean : sigma)                     ... 1.00 : 0.50
Number of atoms                             ... 8
Number of fragments                         ... 1
Flexibility parameter                       ... 1.00
Optimizations per global step               ... 528
Optimizations per worker                    ... 66
*Reached the minimum optimizations per worker [fmax(3 * NAtoms, 45)]!
```

The global minimum found is a D_{4h} planar structure, the same as found on the literature for the Au_8 cluster using other DFT methods [571]:

Fig. 4.9: The lowest energy conformer of the Au_8 cluster on the GFN1-xTB PES.

4.11.7 GOAT-REACT: an algorithm for automatic reaction pathway exploration

Another variant of the GOAT algorithm was created to allow for automatic reaction exploration, which in the end is nothing more than an exploration on the collective PES of reactant and product.

Here the user can be really creative and there are many different ways to explore this algorithm, but let us start with a simple reaction: the gas phase reaction of ethylene and singlet oxygen.

Fig. 4.10: What could be the products of the reaction between ethylene and singlet oxygen?

After running the following input:

```
!XTB GOAT-REACT

* XYZ 0 1
C         -3.26482       -0.47497        0.33191
C         -2.16518        0.24269        0.35382
H         -4.23539       -0.01923        0.27823
H         -3.23979       -1.54754        0.37118
H         -2.19035        1.31540        0.31866
H         -1.19481       -0.21295        0.41157
O         -3.42426       -0.30941        2.30779
O         -2.17088        0.05188        2.32517
*
```

the output looks more or less similar to the regular `GOAT-EXPLORE`, except for a few differences:

1. The maximum barriers GOAT is allowed to cross are higher.
2. The very initial geometry optimization is skipped by default.
3. `AUTOWALL` is set to `TRUE` which means that an ellipsoid wall potential is added using the maximum x,y,z dimensions of the molecule + 5 Angs as radii.

Another important factor is the maximum topological difference (`MAXTOPODIFF`), which is set to 8 by default, as printed on the output:

```
Global parameters
-----------------
  GOAT version                                ... react
  Max. topological diff.                      ... 8
  Minimum global steps                        ... 3
  Number of base workers                      ... 4
```

`MAXTOPODIFF` is a key concept here. If one simply looks for all possible topological permutations between reactants and products, even simple systems such as this could lead to an enormous number of combinations.

We defined the topological difference simply as the sum of broken bonds + formed bonds from some reference structure, which is taken from the structure obtained after the first geometry optimization inside the GOAT iterations (before any uphill step).

There will be more files printed than usual, the most important ones are:

1. `Basename.products.xyz` – contains all *reactomers* and all their conformers for the reaction. It is usually a very large file.
2. `Basename.products.topodiff2.xyz` – contains only those separated with topodiff 2 from the reference structure, and so on.
3. `Basename.products.unique.xyz` – contains a list of all topologically **unique** products, without their conformers or rotamers, only the lowest energy conformer is printed. Here the reference structure will come first and the others will be shown with their relative energy difference in kcal/mol.

Some of the products found as an example for this reaction are:

Fig. 4.11: Some products automatically found by GOAT-REACT using the input given above.

Note

Please be aware that singlet oxygen is so reactive that even the first optimization leads to a cyclization reaction and the reaction product is then taken as the reference structure.

Important

The use of force-fields like the GFN-FF is not recommended here, because it is not supposed to break bonds.

4.11.8 Exploring geometrical diversity: the GOAT-DIVERSITY option

Besides looking for an accurate ensemble around the global energy minimum, in some cases one might be looking only for **structural** diversity instead. For example, if you know your PES is too bad anyways for a given system, it might make sense to explore first the geometrical diversity and only later refine those on a higher level PES optimization.

We have a special keyword for that called `!GOAT-DIVERSITY`, which will:

1. Ignore energy differences when filtering different conformers;
2. Set the RMSD difference value for the filtering from 0.125 to 0.5 Angs;
3. Set the geometry convergence criteria for `SLOPPYOPT` - we are not looking for very accurate energies;
4. Allow conformers up to 60 kcal/mol to be included.

With all that, what effectively happens is that a set of structures within a radius of about 0.5 RMSD from each other are taken into the final ensemble, irrespective of energy differences. This distance can be controlled as usual by:

```
%GOAT RMSD 1.5 END # RMSD here in angstroem
```

Also, if some of the usual optimization flags are set like `!NORMALOPT` or `!TIGHTOPT`, these will overwrite the `SLOPPYOPT` set here automatically.

4.11.9 Automated coarse-grained with GOAT-COARSE

For large systems, we can also now combine GOAT with the `RIGIDBODYOPT` (Section 4.1.4) and the *automatic fragmentation* options to reduce the dimensionality of large system. The single input that controls that is called `!GOAT-COARSE`. Important is that it will **not** define the fragments by itself, that is something you have to do.

For example, you can give phenyl groups, isopropyl, cyclohenane, terc-butyl and etc. as fragments using your own library, and the `GOAT-COARSE` will make sure that these will only rotate and translate as rigid bodies. Or you can split peptides into the individual amino-acids. There is a lot of room to work here!

From the point of view of an algorithm like GOAT, that means reducing the 11 atoms of a phenyl group to essentially 1. Since the conformational search is an exponential-scaling problem, the effective time reduction can be very large, and actually more than a factor of 10.

> **Note**
>
> Currently, for large systems, the rate-limiting step using fast Hamiltonians (e.g. xTB) or force-fields is still the transformations necessary for the internal coordinate transformations. More on the will come with future ORCA versions.

Feel free to be creative with the fragmentation, and as long as the `GOAT-COARSE` flag is there, GOAT will adapt for a significant speedup. This is essentially an automatically coarse-grained version of it.

4.11.10 Some general observations

Default frozen coordinates during uphill step

During the uphill phase only, by default GOAT will freeze:

1. all bonds,
2. all angles involving two sp2 atoms within the same ring,
3. all dihedrals around a strong bond (d(B,C) < [0.9 x (sum of covalent radii)]).

"sp2 atoms" are here loosely defined only for C, N and O with less than 4,3 and 2 bonds respectively.

The first freeze is to avoid change of topology by bond breaking. The second and third are to avoid going over very high energy barriers on changing these angles, which in practice, unless under very special circumstances will never flip anyway!

These constraints are automatically lifted for `GOAT-EXPLORE` and can also be set to `FALSE` with their specific keywords.

Parallelization of GOAT

GOAT will profit from a large number of cores in a different way than most ORCA jobs, because it distributes the necessary work along different workers. It can also work multidone and distribute these workers through different nodes.

Since there is usually a regular first optimization step before starting GOAT, which will **not** profit from a large number of cores, these are limited by the flag `MAXCORESOPT` and set to a maximum of 32. After that, GOAT will switch back to use all cores provided. We do not recommend changing that maximum number, because it will probably only make things slower, but it can be controlled inside the `%GOAT` block.

Tips and extra details

Note

- GOAT will work with any method in ORCA, all you need is the gradient. That includes using DFT, QM/MM, ONIOM, broken-symmetry states, excited states etc.
- Be aware that DFT is **much** costlier than XTB. It is perfectly possible to run GOAT with R2SCAN-3C, but be prepared to use many cores or wait for a few days :D. We recommend at least `%PAL NPROCS 32 END`, to have 8 workers with 4 cores each. Hybrid DFT is even heavier, so if you want to use B3LYP,go with at least `NPROCS 64` - and don't hurry. The aim is to do a global search here, it does not come for free!
- In many cases, it might be useful to use `GFNUPHILL GFNFF` to use the GFN-FF force-field PES during the uphill steps. There, an exact potential is not really needed as the main objective is to take structures out of their current minimum and GOAT will run much faster, only using the chosen method for the actual optimizations. GFN2XTB, GFN1XTB or GFN0XTB are also valid options.
- For methods that need bond breaking, such as `GOAT-EXPLORE` or `GOAT-REACT`, `GFNUPHILL gfnff` cannot be used because the GFN-FF will not allow for bond breaking. Choose GFN2-xTB (`gfn2xtb`), GFN1-xTB (`gfn1xtb`) or GFN0-xTB (`gfn0xtb`).
- You can always check what the workers are doing by looking into the `Basename.goat.x.x.out` files. The first number refers to the global iteration and the second to the specific worker. This is an ORCA output (with some suppressed printing to save space) that can be opened in most GUIs.
- GOAT automatically enables fragment detection at the very beginning of the process, even before the first geometry optimization. It also respects fragments defined via the `%frag` block. For a detailed explanation of fragment selection, see the *Fragment Specification* section.
- Amide bond chirality is **not** frozen by default, which means the input topology you gave for amides (cis or trans) may change. If you want to freeze it, set `FREEZEAMIDES` to `TRUE`.
- Similarly double bonds outside rings can also change their topology. Choose `FREEZECISTRANS TRUE` in order to freeze those dihedrals.
- For certain molecules, it might be interesting to limit the coordination number of certain atoms, in that case use `MAXCOORDNUMBER`.
- GOAT will respect the choices from the `%GEOM` block for the geometries so you can use all kinds of constraints you need for other types of coordinate freeze. It can also be combined with all kinds of arbitrary bias and wall potentials available (see Section 4.1.5).
- If you want to push only certain atoms uphill, you can give a list to `UPHILLATOMS`. In that case the uphill force shown in Fig. 4.6 will be applied only to the coordinates involving those atoms and the rest of the molecule will only react to that. This is useful for conformational searches on parts of a bigger system.
- By default conformers up to 12.0 kcal/mol from the global minimum are included, this can be changed by setting `MAXEN`.

4.11.11 Keywords

A collection of GOAT related simple input keywords is given in Table 4.8. All `%goat` block options are summarized in Table 4.9.

Table 4.8: Simple input keywords for the GOAT algorithm.

Keyword	Description
`GOAT`	Request a default *GOAT* run
`GOAT-ENTROPY`	Request a *GOAT-ENTROPY* run
`GOAT-EXPLORE`	Request a *GOAT-EXPLORE* run
`GOAT-REACT`	Request a *GOAT-REACT* run
`GOAT-DIVERSITY`	Request a *GOAT-DIVERSITY* run
`GOAT-COARSE`	Request a *GOAT-COARSE* run

Table 4.9: `%goa`

Keyword	Options	Description
`MAXITER`	`128`	defines an arbitrary number of max GOAT geom
`MAXOPTITER`	`256`	maximum number of geometry optimizations per
`SKIPINITIALOPT`	`true`	if you want to skip the initial optimization (defaul
`RANDOMSEED`	`true`	set it to false to have a deterministic GOAT run. s
`READENSEMBLE`	`"name.xyz"`	an ensemble file to be read. the comment line sho
`AUTOWALL`	`true`	automatically create an ellipsoid wall potential arc
`TEMPLIST`	`3000, 2000, 750, 500`	a list of temperatures, defines the number of basic
`MAXCORESOPT`	`32`	the max. number of cores used during the very fir
`PRINTINTERNALS`	`true`	print all the internal coordinates on the worker ou
Worker options:		
`NWORKERS`	`auto`	define the number of workers (default auto). auto
`MAXITERMULT`	`3`	a simple keyword to multiply the number of geom
`MINGLOBALITER`	`3`	the minimum number of global iterations needed
`KEEPWORKERDATA`	`false`	set to true to keep the worker outputs (might be a
`WORKERRANDOMSTART`	`true`	after the first cycle, each worker starts with a rand
Uphill step:		
`UPHILLATOMS`	`{0:2 5 14:29} end`	if given, only those atoms listed will be pushed, u
`GFNUPHILL`	`gfnff`	use GFN-FF (`gfnff`) only during the uphill step
Filtering and screening:		
`ALIGN`	`false`	align all final conformers with respect to the lowes
`ENDIFF`	`0.1`	minimum energy difference needed to differentiat
`MAXEN`	`6.0`	the maximum relative energy of a conformer to b
`RMSD`	`0.125`	minimum RMSD to differentiate conformers, in A
`ROTCONSTDIFF`	`0.01`	maximum difference for the rotational constant, i
`RMSDMETRIC`	`eigenvalue`	use eigenvalues of distance matrix for RMSD? de
`BONDFACTOR`	`1.2`	the multiplier for the sum of covalent radii which
GOAT-ENTROPY:		
`MAXENTROPY`	`false`	add delta Gconf as convergence criteria (default fa
`CONFTEMP`	`298.15`	temperature used to compute the free energy, in K
`MINDELS`	`0.1`	the minimum entropy difference between two iter
`CONFDEGEN`	`2, auto, automax`	set an arbitrary degeneracy per conformer? auto f
Free topology:		
`FREEHETEROATOMS`	`false`	free all atoms besides H and C.
`FREENONHATOMS`	`false`	free all non H atoms
`FREEFRAGMENTS`	`false`	free interfragment topology, i.e., bonds between f
`TOPOBREAK`	`1,5,6,13`	a list of atom for which the topology *might* break
Freeze options (we don't recommend changing these unless you really need to!):		
`FREEZEBONDS`	`false`	freeze bonds uphill (default true)
`FREEZEANGLES`	`false`	freeze sp2 angles and dihedrals uphill (default true
`FREEZECISTRANS`	`false`	freeze cis-trans isomers outside rings (default false
`FREEZEAMIDES`	`false`	freeze amide cis/trans chirality (default false)
`MAXCOORDNUMBER`	`10, 4, 11, 6`	a list of "atom number, coordination number" tha
GOAT-REACT:		
`MAXTOPODIFF`	`8`	the maximum topological difference that is allow.

4.12 SOLVATOR: Automated Explicit Solvation

The SOLVATOR is an algorithm to automatically add explicit solvent molecules to a given system. By default, SOLVATOR makes use of the *DOCKER* algorithm. If a very large number of solvent molecules should be added, a very fast but less accurate *stochastic method* method is available. The *ALPB solvation model* is used to create an implicit continuum solvation around the generated clusters. Currently the SOLVATOR is **only** working with the *GFN-xTB and GFN-FF methods*. A complete list of available keywords and options can be found in Section 4.12.8.

> **Note**
>
> The DOCKER on ORCA 6.1 was significantly improved, in both speed and accuracy, when compared to ORCA 6.0. These changes can also alter the results of the SOLVATOR, but only because they will be even more accurate.

4.12.1 Basic Usage

The default SOLVATOR is invoked via the `%SOLVATOR` block by defining the number of solvent molecules via `NSOLV <integer>`. The solvent to be added can be either chosen from ORCA's internal solvent library in combination with the `!ALPB(<solvent>)` simple input keyword (cf. Table 3.24) or by providing a *custom solvent*.

> **Important**
>
> - The SOLVATOR will by default make use of `CLUSTERMODE DOCKING` with settings that are equivalent to `!QUICKDOCK` (cf. *DOCKER*). For more accurate results `!NORMALDOCK` or even `!COMPLETEDOCK` may be used, however these approaches will be much slower!
> - All options related to the docking process can be controlled via the `%DOCKER` block as described in Section 4.13.
> - All options given in the `%GEOM` block such as *constraints* etc., will be respected during the docking of the solvent, but can only be given for the solute.

```
!GFN2-XTB ALPB(<solvent>)

%SOLVATOR
   NSOLV <integer>
END
```

A simple example could be adding three water molecules to a Histidine molecule using the default settings and the GFN2-xTB method.

```
!GFN2-XTB ALPB(WATER) PAL16

%SOLVATOR
   NSOLV 3
END

* XYZ 0 1
  N     0.885996   -0.961304   -0.120339
  C     1.798313    0.104987    0.275069
  C     1.249714    0.744242    1.567548
  O     1.573032    1.831447    1.951866
  C     2.049102    1.187937   -0.781297
  C     2.714441    0.645674   -1.999072
  N     2.728606    1.335092   -3.185194
  C     3.401683    0.571600   -4.081842
  N     3.805612   -0.545068   -3.552995
```

(continues on next page)

(continued from previous page)

```
  C     3.389082   -0.516790   -2.258890
  O     0.397674   -0.041862    2.212628
  H     0.272440   -0.853698    1.671197
  H     1.339440   -1.612664   -0.750009
  H     0.086389   -0.572348   -0.612909
  H     2.756495   -0.353470    0.548654
  H     2.661849    1.969568   -0.321790
  H     1.092545    1.646672   -1.057454
  H     2.328734    2.246496   -3.338113
  H     3.566873    0.866218   -5.098496
  H     3.616501   -1.333139   -1.602802
*
```

The output report the solvent chosen, together with some details about its dimensions, the number of molecules to be added and the method. The structure from the internal database is also always printed.

```
                        *****************
                        * ORCA Solvator *
                        *****************


Solvent chosen:                                 WATER
Solvent radius:                                 .... 1.69 Angs
Solvent max dimensions (x,y,z):                 .... 2.73, 2.32, 1.52 Angs
Number of solvent molecules to be added:        .... 3 molecules
Method used to add the solvent:                 .... docking
Number of atoms of solvent molecule:            .... 3 atoms
Coordinates of solvent in Angstroem:

   O    0.000014   0.401429   0.000000
   H    0.765192  -0.200729   0.000000
   H   -0.765206  -0.200700   0.000000

Solute radius:                                  .... 5.27 Angs
Ellipsoid potential radii:                      .... 8.37, 6.28, 5.76 Angs
```

Further, the solvation process is monitored per solvent molecule and the final result is printed to the file `basename.solvator.xyz`. There will be also an intermediate file named `basename.solvator.solventbuild.xyz` with the solvent molecules added one by one. On the output `Einter` is the interaction energy obtained from the DOCKER used within the SOLVATOR process and `dE` is the different between the current and the previous `Einter`.

```
Adding solvent molecules to the solute         ....


      Iter       Energy      Einter          dE          Time
                  (Eh)     (kcal/mol)    (kcal/mol)     (min)
      -----------------------------------------------------
         1    -39.452446    -4.331670     -4.331670      0.26
         2    -44.544347    -4.316639      0.015031      0.34
         3    -49.636018    -4.171990      0.144649      0.40

Final radius after microsolvation:             .... 4.88 Angs
Time needed for microsolvation   :             .... 60.54 s
Final structured saved to        :             HIS.solvator.xyz

                ****ORCA-SOLVATOR TERMINATED NORMALLY****
```

Note

In contrast to the *DOCKER*, the solute is frozen by default in the SOLVATOR algorithm. Set `FIXSOLUTE`

FALSE in the %SOLVATOR block to change that.

The solvated Histidine molecule generated this way is depicted in Fig. 4.12.

Fig. 4.12: Three water molecules added to Histidine by the SOLVATOR.

4.12.2 Custom Solvents

The SOLVATOR method itself is agnostic to the solvent, and in principle any other solvent can be used. If the desired solvent is not available in ORCA's *internal solvent database*, a custom solvent molecule can be provided in .xyz format via the SOLVENTFILE keyword in the %SOLVATOR block.

```
%SOLVATOR
   SOLVENTFILE "solvent_file_name.xyz"
END
```

As with the *DOCKER*, the charge and multiplicty can be given to the solvent as two integers on the comment line of the solvent_file_name.xyz (default is neutral closed-shell). Since the ALPB method only has a limited number of *available solvents*, the the dielectric constant ϵ should be approximated to the next closest available solvent. Currently, it is not possible to define custom dielectric constants for ALPB.

For example, isopropanol can be used as solvent by providing a isopropanol.xyz file. In this case, the dielectric constant of isopropanol is approximated to that of ethanol. As the ALPB implicit solvation is a quite crude implicit solvation model the results will be sufficiently similar. The resulting cluster is depicted in Fig. 4.13.

```
12
0 1
C          -3.79410        2.24670       -0.09622
C          -3.45574        0.76660       -0.18820
H          -2.94382        2.85306       -0.42645
H          -4.00575        2.53923        0.93817
H          -4.66172        2.49512       -0.71492
C          -4.60559       -0.10847        0.28691
H          -4.84802        0.09429        1.33594
H          -4.32886       -1.16657        0.22730
H          -5.50341        0.05251       -0.31744
O          -2.30376        0.49878        0.60542
H          -3.20686        0.51239       -1.22373
H          -2.51694        0.72259        1.52758
```

```
!GFN2-XTB ALPB(ETHANOL) PAL16

%SOLVATOR
   SOLVENTFILE "isopropanol.xyz"
   NSOLV 3
END

* XYZFILE 0 1 HIS.xyz
```

Fig. 4.13: Three iso-propanol molecules added by the solvator.

4.12.3 Stochastic Method

In case you want to add a really large number of explicit solvent molecules, the `STOCHASTIC` mode will be significantly faster. It can be invoked via the `CLUSTERMODE STOCHASTIC` keyword in the `%SOLVATOR` block.

> **Warning**
>
> `CLUSTERMODE STOCHASTIC` is a probabilistic approach and is not nearly as accurate as `CLUSTERMODE DOCKING`! Nonetheless it is useful for quite a few applications.

```
!GFN2-XTB ALPB(WATER) PAL16

%SOLVATOR
    NSOLV       100
    CLUSTERMODE STOCHASTIC
END

* XYZFILE 0 1 HIS.xyz
```

The output will slightly differ from that generated by default. For this example, hundred water molecules are added in seconds to the histidine molecule (Fig. 4.14).

```
Solvent chosen:                                 WATER
Solvent radius:                                 .... 1.69 Angs
Solvent max dimensions (x,y,z):                 .... 2.73, 2.32, 1.52 Angs
Number of solvent molecules to be added:        .... 100 molecules
Method used to add the solvent:                 .... stochastic
Number of atoms of solvent molecule:            .... 3 atoms
Coordinates of solvent in Angstroem:
```

(continues on next page)

(continued from previous page)

```
   O    0.000014   0.401429   0.000000
   H    0.765192  -0.200729   0.000000
   H   -0.765206  -0.200700   0.000000

Solute radius:                                  .... 5.27 Angs
Adding solvent molecules to the solute          ....

      Iter     Target function   Time
                  (Coulomb)     (min)
      -------------------------------
         1     -4.659435e-07     0.00
         2     -4.604606e-07     0.00
         3     -2.519684e-07     0.00
      (...)

Final radius after microsolvation:              .... 9.79 Angs
Time needed for microsolvation   :              .... 5.11 s
Final structured saved to        :              HIS_STOCHASTIC.solvator.xyz

                ****ORCA-SOLVATOR TERMINATED NORMALLY****
```

Fig. 4.14: A hundred water molecules added to Histidine using `CLUSTERMODE STOCHASTIC`.

Details on the Stochastic Method

`CLUSTERMODE STOCHASTIC` basically uses random trial and error to assign the placing of the solvents. The algorithm actually uses information from self-consistent EEQ charges [159] as part of an simplistic potential to guide the placing of polar molecules in a more reasonable way.

After trying to distribute the solvent molecule somewhere and checking for clashes, we first compute the electrostatic energy between solvent and solute:

$$E_{elec} = \sum_{A}^{solute} \sum_{B}^{solvent} \frac{Q_A Q_B}{R_{AB} 4\pi\epsilon_{solv}}, \tag{4.30}$$

and define our target function to be minimized (V) as a damped version of that, using the minimum distance R_{min} between the solute and solvent atoms:

$$\begin{aligned} V &= E_{elec} e^{-R_{min}^2} && \text{if } E_{elec} < 0 \\ &= R_{min} && \text{otherwise,} \end{aligned} \tag{4.31}$$

The `STOCHASTIC` mode then consists of finding the correct solvent placement that minimizes V for a given solute. The damping function is there to ensure that:

1. The electrostatic interaction decays strongly with distance,
2. Repulsive energies will be so unfavorable that only the distance will matter.

The result is such that solvent molecules are placed as close as possible to the solute and maximizing electrostatic interactions. This helps to create the solvent shell such that is does look like the actual result one would expect from a more elaborate calculation, but with essentially zero cost.

The best value for V found after each solvent was added is what is printed in the output as `Target function`:

```
Iter     Target function   Time
                          (min)
------------------------------
   1     -4.342597e-07     0.00
   2     -3.166857e-07     0.00
   3     -4.814590e-08     0.00
```

When using the *`DROPLET` mode* , the R_{min} is defined as the distance to the centroid of the solute, instead of that of the closest atom pair instead.

If you don't want to include the electrostatic component for any reason, just set `%SOLVATOR USEEEQCHARGES FALSE END`. In the future other charge models will be available as well.

4.12.4 Creating a Droplet

Warning

The droplet mode and its variants are only available for *`CLUSTERMODE STOCHASTIC`*.

The regular stochastic method will create a solvation sphere around the solute following its shape and topology. If required, the `DROPLET` mode can be activated to create spherical droplets whose sizes are controlled either by the number of solvent molecules added via `NSOLV` or by a predefined radius controlled via the `RADIUS` keyword. If a predefined radius is given, the solvator will add solvent molecules until the desired radius is reached.

Spherical droplets are invoked by the `DROPLET TRUE` keyword. If the size is defined by `NSOLV` the resulting cluster will look like Fig. 4.15.

```
!XTB ALPB(WATER) PAL16

%SOLVATOR
    NSOLV       100
    CLUSTERMODE STOCHASTIC
    DROPLET     TRUE
END

* XYZFILE 0 1 HIS.xyz
```

Fig. 4.15: Hundred water molecules added around histidine by the SOLVATOR enforcing spherical symmetry.

If the maximum radius is defined by `RADIUS` the SOLVATOR will add as many molecules as necessary until the radius is reached. The radius is always taken from the centroid of the solute.

```
!GFN2-XTB ALPB(WATER) PAL16

%SOLVATOR
    RADIUS      15 # in angstroem
    CLUSTERMODE STOCHASTIC
END

* XYZFILE 0 1 HIS.xyz
```

For this example, the output print the actual droplet radius reached and the number of molecules required to reach it, in this case 668. The resulting cluster is depicted in Fig. 4.16

```
Desired sovent radius:                          .... 15.00 Angs
Actual sovent radius:                           .... 15.18 Angs
Final number of solvent molecules:              .... 668 molecules
```

Fig. 4.16: A droplet created with 668 water molecules to achieve a radius of approximately 15 Angstroem.

4.12.5 The Wall Potential

If one uses the default approach using the `CLUSTERMODE DOCKING`, a fictitious wall potential is added to guarantee that the solvents are added such that they fill most of the first solvation sphere around the solute before being placed further.

> **Note**
>
> This resembles to some extent what was published recently by the group of Prof. Stefan Grimme (called "quantum cluster growth"), but here only a single outer wall potential is used [572]. Otherwise, the present algorithm is independent and unrelated to it.

As one can see from the output of the Histine example discussed in the previous sections, by default an ellipsoid potential is built with dimensions such that it will enclose the solute plus at least one molecule of the solvent in all directions (Fig. 4.17).

```
Ellipsoid potential radii:                    .... 8.37, 6.28, 5.76 Angs
```

Fig. 4.17: A simple scheme to show how the wall potential is built to keep the solvent molecules close to the solute space.

A single parameter controlled by `SOLVWALLFAC` in the `%SOLVATOR` block defines how further this wall is built outside the solute. Its default value is 1.0, and increasing it to larger values will increase the default initial wall by about half the sum of the maximum dimensions of solute plus solvent for each unit.

The initial wall potential is by default not changed, unless a) the algorithm can not place a solvent, or b) the energy of the placed solvent is higher than before. Only then the walls will be updated to help allocating the next solvent molecule, and a message will be printed with the current scaling factor.

4.12.6 Controlling Randomness

Both clustermodes, `DOCKER` and `STOCHASTIC`, are intrinsically dependent on random numbers. However, ORCA sets a fixed random seed such that the same results are always obtained on the same machine if calculations are repeated.

In order to make both fully random, please use the `RANDOMSOLVATION TRUE` keyword in the `%SOLVATOR` block.

4.12.7 Vacuum Search

One option that might come in handy under certain conditions is to use the SOLVATOR to add the explicit solvent based on the implicit solvation information, but actually **not** use any implicit solvation while trying to place them.

That is useful for instance when trying to generate aggregates of solute plus solvents that can form in gas-phase only, where there will be no other solvent molecules around. That can be set by the `VACUUMSEARCH TRUE` keyword in the `%SOLVATOR` block and the implicit solvation method will be ignored to compute energies and gradients. In the `STOCHASTIC` method, the ϵ_{solv} will be set to 1.0.

Important

By the time of this release, this algorithm was still not published. Publications are under preparation.

4.12.8 Keywords

A collection of SOLVATOR related keywords can be fount in Table 4.10.

Table 4.10: `%SOLVATOR` block input keywords for the SOLVATOR algorithm.

Keyword	Options	Description
`NSOLV`	`10`	number of explicit solvent molecules to be added.
`SOLVENTFILE`	`"solvent_file_name.xyz"`	a file for custom solvents. NOT needed for the regular solvents available via ALPB(solvent). charge and multipl. can be given on the comment.
`CLUSTERMODE`	`docking`	Docking method for adding new solvent molecules (default).
	`stochastic`	Stochastic method for adding new solvent molecules. Default is `docking`.
`PRINTLEVEL`	`low`	Low level of print verbosity. Default is `normal`.
	`normal`	Normal level of print verbosity (default).
	`high`	High level of print verbosity. Default is `normal`.
`RANDOMSOLV`	`false`	Make it completely random (default = `false`) .
`FIXSOLUTE`	`true`	keep the solute constrained? (default = `true`)
`VACUUMSEARCH`	`false`	Activate the search in vacuum. (default = `false`)
Stochastic method:		
`USEEEQCHARGES`	`true`	Use eeq charges during the stochastic mode? Default is true (default = `true`).
`DROPLET`	`false`	Create a spherical droplet (default = `false`).
`RADIUS`	`10`	A radius in Angström for the droplet. Solvent molecules will be added until the target radius is reached.
Docking method:		
`WALLFAC`	`1.0`	Factor use to define the initial size of the wall potential.

Note

All other docking options are controlled as usual via the `%docker` block. Flags such as `!NORMALDOCK` and `!COMPLETEDOCK` will apply here as well.

4.13 DOCKER: Automated Docking Algorithm

The most important aspects of chemistry/physics do not occur with single molecules, but when they interact with each other. Now, given any two molecules, how to put them together in the best interacting "pose"? That is what we try to answer when using the ORCA DOCKER. Docking here refers to the process of taking two systems and putting them together in their best possible interaction.

Note

The DOCKER on ORCA 6.1 was significantly improved, in both speed and accuracy, when compared to ORCA 6.0. These changes can also alter the results, but only because they will be even more accurate.

4.13.1 Example 1: A Simple Water Dimer

Let us start with a very simple example. Given two water molecules, how to find the optimal dimer? With the DOCKER that is simple and can be done with:

```
!XTB

%DOCKER GUEST "water.xyz" END

* xyz 0 1
O  -2.13487    2.63905   -0.01809
```

(continues on next page)

(continued from previous page)

```
H  -1.16698    2.61938    0.02397
H  -2.41372    2.24598    0.82256
*
```

where the file `water.xyz` is a `.xyz` file which contains the same water structure, optionally with charge and multiplicity (in that order) on the comment line (the second line by default):

```
3
0 1
O  -2.13487    2.63905   -0.01809
H  -1.16698    2.61938    0.02397
H  -2.41372    2.24598    0.82256
```

The molecule given on the regular ORCA input will be the `HOST`, and the `GUEST` is always given through an external file.

The output will start with:

```
                                ***************
                                * ORCA Docker *
                                ***************

Reading guests from file             water.xyz
Number of structures read from file  1
Charge and multiplicity of guest     from file
Docking approach                     independent
Docking level                        normal
Optimizing host                      .... -5.070544 Eh
Optimizing guest                     .... -5.070544 Eh
```

where it writes the name of the file with the `GUEST` structure, the number of structures read, some extra info and will optimize both host and guest (in this case they are the same), here by default using GFN2-XTB.

> **Note**
>
> If no multiplicity or charge are given, the `GUEST` is assumed to be neutral and closed-shell.

> **Note**
>
> The DOCKER right now is **only** working with the GFN-XTB and GFN-FF methods and the ALPB solvation model. It will be expanded later to others.

That is followed by some extra info that is explained in more details on its own detailed section (see *DOCKER: Automated Docking Algorithm*):

```
Starting Docker
---------------
Guest structure                      .... structure number 1
Guest charge and multiplicity        .... (0 , 1)
Final charge and multiplicity        .... (0 , 1)
PES used for swarm intelligence      .... GFN2-XTB
Setting random seed                  .... done
Creating spatial grid
   Grid Max Dimension                5.50 Angs
   Angular Grid Step                 32.73 degrees
   Cartesian Grid Step               0.50 Angs
   Points per Dimension              11 points
```

(continues on next page)

(continued from previous page)

```
Initializing workers
   Population Density                0.50 worker/Ang^2
   Population Size                   57
Swarm intelligence search
   Minimization Algorithm            mutant particle swarm
   Min, Max Iterations               (3 , 10)
```

That is followed by the docking itself, which will stop after a few iterations:

```
   Iter   Emin          avDE         stdDE            Time
          (Eh)       (kcal/mol)    (kcal/mol)        (min)
   -------------------------------------------------------

       1 -10.147462     2.756033     1.821981         0.03
       2 -10.147462     2.121389     1.610208         0.03
       3 -10.148583     2.313606     1.365227         0.03
       4 -10.148583     1.846998     1.188680         0.02
       5 -10.148583     1.587332     1.168207         0.02
No new minimum found after 3 (MinIter) steps.
```

The idea here is to collect as many local minima as possible, that is, collect a first guess for all possible modes of interaction between the different structures. We do this by allowing both structures to partially optimize, but it is important to say we will not look for multiple conformers of the host and guest here.

With all solutions collected, we will take a fraction of them and do a final full optimization:

```
Running final optimization
   Maximum number of structures      7
   Minimum energy difference         0.10 kcal/mol
   Maximum RMSD                      0.25 Angs
   Optimization strategy             regular
   Coordinate system                 redundant 2022
   Fixed host                        false

   Struc   Eopt       Interaction Energy       Time
           (Eh)               (kcal/mol)      (min)
   -----------------------------------------------

    1      -10.149006          -4.968378       0.01
    2      -10.149005          -4.967965       0.01
    3      -10.149007          -4.968825       0.01
    4      -10.149007          -4.968641       0.01
    5      -10.149007          -4.968743       0.01
    6      -10.149006          -4.968116       0.01
    7      -10.149007          -4.968678       0.01
```

And as you can see, we also automatically print the `Interaction Energy`, which is simple an energy difference between the final complex, host and guest. The final best structure with lowest interaction energy is then saved on the `Basename.docker.xyz` file. If needed, all other structures are saved on the `Basename.docker.struc1.all.optimized.xyz`, as written on the output:

```
All optimized structures saved to  :      Basename.docker.struc1.all.optimized.xyz

-------------------------------------
LOWEST INTERACTION ENERGY:  -4.968825 kcal/mol (structure 3)
-------------------------------------

(...)

The lowest energy structure was 1, with energy -10.149007.
Docked structures saved to    Basename.docker.xyz
```

> **Note**
>
> The name `Basename.docker.struc1.all.optimized.xyz` refers to `struc1` because that is the first docked guest. Later that can be done with multiple guest and that is only a way to organize the outputs.

We are all set, the output can be visualized and it is, as expected:

Fig. 4.18: The final water dimer found using the GFN2-XTB PES.

4.13.2 Example 2: A Uracil Dimer

Now for a slightly more complex example, a uracil dimer:

```
! XTB PAL16

%DOCKER GUEST "uracil.xyz" END

*xyz 0 1
N    -0.2707028    0.7632994    1.0276159
H    -0.5957915    1.3097757    1.8163465
C    -0.3386212    1.3810817   -0.2276640
O    -0.7270425    2.5346295   -0.3329857
N     0.3638189   -1.2896563    0.1949192
H     0.0796815    0.9143946   -2.3190044
C     0.3781329   -0.7736192   -1.0714063
H     0.6499130   -1.4675080   -1.8526542
C     0.0669084    0.5154897   -1.3194961
H     0.4818502   -2.2779688    0.3498201
C    -0.0016589   -0.5616540    1.3117092
O    -0.0864879   -1.0482643    2.4227999
*
```

where the `uracil.xyz` is a simple repetition of the structure, as with the water before.

In this case the output is more diverse, and in fact many different poses appear as candidates for the final optimization:

```
   Struc    Eopt        Interaction Energy        Time
            (Eh)                (kcal/mol)       (min)
   ------------------------------------------------

    1      -49.248577          -11.723457        0.08
    2      -49.250442          -12.893758        0.08
    3      -49.245624           -9.870339        0.03
    4      -49.252991          -14.493130        0.06
```

(continues on next page)

(continued from previous page)

```
    5      -49.248470         -11.656256       0.05
    6      -49.259335         -18.474228       0.05
    7      -49.259269         -18.432902       0.08
    8      -49.254913         -15.699019       0.03
    9      -49.254927         -15.708244       0.03
   10      -49.241672          -7.390198       0.02
   11      -49.246534         -10.441269       0.03
```

and structure number 6 is found to be the one with lowest interaction energy:

```
-------------------------------------
LOWEST INTERACTION ENERGY: -18.474228 kcal/mol (structure 6)
-------------------------------------
```

Here is a scheme with the structures found and their relative energies:

Fig. 4.19: Uracil dimer structures generated by DOCKER (duplicates removed) with relative energies in kcal/mol.

Note

There might be duplicated results after the final optimization, these are currently **not** automatically removed. Here they were manually removed.

Important

The `PAL16` flag means XTB will run in parallel, but the ORCA DOCKER could be parallelized in a much more efficient way by paralleizing over the workers. That will be done for the next versions and it will be significantly more efficient.

4.13.3 Example 3: Adding Multiple Copies of a Guest

Suppose you want to add multiple copies of the same guest, for example three water molecules on top of the uracil one after the other. That can be simply done with:

```
! XTB PAL16

%DOCKER
    GUEST        "water.xyz"
    NREPEATGUEST 3
END

*xyz 0 1
N    -0.2707028    0.7632994    1.0276159
H    -0.5957915    1.3097757    1.8163465
C    -0.3386212    1.3810817   -0.2276640
O    -0.7270425    2.5346295   -0.3329857
N     0.3638189   -1.2896563    0.1949192
H     0.0796815    0.9143946   -2.3190044
C     0.3781329   -0.7736192   -1.0714063
H     0.6499130   -1.4675080   -1.8526542
C     0.0669084    0.5154897   -1.3194961
H     0.4818502   -2.2779688    0.3498201
C    -0.0016589   -0.5616540    1.3117092
O    -0.0864879   -1.0482643    2.4227999
*
```

and the guests on `water.xyz` will be added on top of the previous best complex three times. Now, there will be files with names `Basename.docker.struc1.all.optimized.xyz`, `Basename.docker.struc2.all.optimized.xyz` and `Basename.docker.struc3.all.optimized.xyz`, one for each step. Still, the same final `Basename.docker.xyz` file and now a `Basename.docker.build.xyz` is also printed, with the best result after each iteration.

That's how the results look like, from the `Basename.docker.xyz`:

Fig. 4.20: Cumulative docking of three guests

> **Note**
>
> By default the `HOST` is always optimized. It can be changed with `%DOCKER FIXHOST TRUE END`.

4.13.4 Example 4: Find the Best Guest

Another common case would be: given a list of guests - or conformers of the same guest (see *GOAT: global geometry optimization and ensemble generator*) - one might want to know what is the "best guest", that is the one with the lowest interaction energy.

In order to do that, simply pass a multixyz file and the DOCKER will try to dock all structures from that file, one by one:

```
! XTB

%DOCKER GUEST "uracil_water.xyz" END

*xyz 0 1
N    -0.2707028    0.7632994    1.0276159
H    -0.5957915    1.3097757    1.8163465
C    -0.3386212    1.3810817   -0.2276640
O    -0.7270425    2.5346295   -0.3329857
N     0.3638189   -1.2896563    0.1949192
H     0.0796815    0.9143946   -2.3190044
C     0.3781329   -0.7736192   -1.0714063
H     0.6499130   -1.4675080   -1.8526542
C     0.0669084    0.5154897   -1.3194961
H     0.4818502   -2.2779688    0.3498201
C    -0.0016589   -0.5616540    1.3117092
O    -0.0864879   -1.0482643    2.4227999
*
```

Here the file `uracil_water.xyz` looks like:

```
3
0 1
O  -2.13487    2.63905   -0.01809
H  -1.16698    2.61938    0.02397
H  -2.41372    2.24598    0.82256
12
0 1
N    -0.2707028    0.7632994    1.0276159
H    -0.5957915    1.3097757    1.8163465
C    -0.3386212    1.3810817   -0.2276640
O    -0.7270425    2.5346295   -0.3329857
N     0.3638189   -1.2896563    0.1949192
H     0.0796815    0.9143946   -2.3190044
C     0.3781329   -0.7736192   -1.0714063
H     0.6499130   -1.4675080   -1.8526542
C     0.0669084    0.5154897   -1.3194961
H     0.4818502   -2.2779688    0.3498201
C    -0.0016589   -0.5616540    1.3117092
O    -0.0864879   -1.0482643    2.4227999
```

with a water followed by an uracil molecule. First, the water will be added, then the uracil, but both separately. The initial output is a bit different:

```
                                 ***************
                                 * ORCA Docker *
                                 ***************


Reading guests from file             uracil_water.xyz
Number of structures read from file  2
Charge and multiplicity of guests    from file
Docking approach                     independent
Docking level                        normal
```

with now two structures being read from file, and the `Docking approach` is labeled as `independent`, meaning each structure will be docked independently of each other.

After everything, the output is:

```
-------------------------------------
LOWEST INTERACTION ENERGY: -18.482854 kcal/mol (structure 6)
-------------------------------------

Total time for docking:       4.84 minutes

The lowest energy structure was 2, with energy -49.259349.
Docked structures saved to    Basename.docker.xyz
```

and one can see that the lowest interaction energy was that of structure 2 (the uracil), meaning it interacts strongly with the `HOST` than the water molecule given. Now the file `Basename.docker.xyz` will contain all final structures, ordered by interaction energy.

Fig. 4.21: Independent docking of water and uracil on top of an uracil molecule

Note

By default, the docking approach uses a fixed random seed and should always give the same result on the same machine. To make it always completely random add `%DOCKER RANDOMSEED TRUE END`.

Note

In order to use the faster GFN-FF instead of GFN2-XTB, use `!DOCK(GFNFF)`. For a quicker (and less accurate) docking, use `!QUICKDOCK`.

Note

To try multiple conformers of the `GUEST`, the ensemble file printed by GOAT `Basename.finalensemble.xyz` can be directly given here and the whole ensemble will be tested against a give `HOST`.

A detailed summary of the other options can be found on *Keywords*.

4.13.5 Underlying theory

The basic idea behind the DOCKER is to use a type of swarm intelligence [573] to find local minima for where the HOST would best be positioned, based on the energies and gradients of a given Potential Energy Surface (PES).

Swarm intelligence algorithms were invented trying to mimic the behavior of animals such as bees and ants, and it actually fits the problem here quite well (as shown below). The basic steps needed for the algorithm are:

1. Optimize HOST and GUEST;
2. Create a spacial grid around the HOST;
3. Initialize a possible set of random solutions;
4. Let the swarm intelligence find local minima;
5. Preoptimize with GFN-FF a large number of these minima;
6. Collect a fraction of the best solutions found on 4. and fully optimize them;
7. The structure with the lowest energy is considered the best solution.

To quickly demonstrate how the initial set of solutions "evolve" to find different local minima, take as an example the substituted biphenyl as HOST below. The figure below demonstrates how a water molecule (GUEST) is docked onto the initial HOST.

Each gray ball represents one possible placing of the water molecule (ignoring here its rotation angle). At the start, they are all distributed over the HOST without any bias. As the iterations go by, the algorithm starts to detect that both sides of the molecule have lower energy solutions than the aromatic rings, as expected, and the possible solutions start to converge to those points.

In the end, solutions containing water molecules binding to both sides are taken, the system is then fully optimized and one finds that the best biding occurs with the hydroxy group to the right.

Fig. 4.22: The docking process of a water molecule on the substituted biphenyl.

Important

Right now the DOCKER can only be used together with the fast GFN-XTB methods, it will be generalized later. It is also not fully exploring the parallelization potential yet, and can be much faster for the next versions.

4.13.6 Looking Deeper into the Output

The details related to the grid and the swarm optimization can be found on the output, here is one example taken from the water dimer example from the earlier section:

```
Creating spatial grid
   Grid Max Dimension                 5.50 Angs
   Angular Grid Step                  32.73 degrees
   Cartesian Grid Step                0.50 Angs
   Points per Dimension               11 points
Initializing workers
   Population Density                 0.50 worker/Ang^2
   Population Size                    57
Swarm intelligence search
   Minimization Algorithm             mutant particle swarm
   Min, Max Iterations                (3 , 10)
```

The `Population Density` here defines how many solutions will be placed on the initial grid around the `HOST` and can be controlled by `%DOCKER SWARMPOPDENSITY 0.50 END`, while the population size is a direct consequence of that, unless specified by `SWARMPOPSIZE` under `%DOCKER`.

Another important piece of information is the `Min, Max Iterations`, which define the minimum number of iterations before checking for convergence, and the maximum possible number of iterations. These can be changed by setting `SWARMMINITER` and `SWARMMAXITER` under the `%DOCKER`.

Then during the swarm search itself (here called the "evolution step"), the `HOST` and `GUEST` are partially optimized using the same criteria as from `!SLOPPYOPT`, and their energy at convergence is taken as a convergence criteria. The printing, still for the previous example, reads:

Note

All of the optimizations respect constraints and/or wall potentials given on the `%GEOM` block. The `HOST` can be easily frozen with `%DOCKER FIXHOST TRUE END`.

```
   Iter    Emin          avDE         stdDE            Time
           (Eh)       (kcal/mol)    (kcal/mol)        (min)
   -----------------------------------------------------

       1 -10.147462     2.756033     1.821981         0.03
       2 -10.147462     2.121389     1.610208         0.03
       3 -10.148583     2.313606     1.365227         0.03
       4 -10.148583     1.846998     1.188680         0.02
       5 -10.148583     1.587332     1.168207         0.02
No new minimum found after 3 (SwarmMinIter) steps.
```

Here we have the number of iterations, the energy of the best solution found during the process, the average energy difference from the lowest solutions and its standard deviation. The first number shows if the system is evolving towards a lower energy solutions and the later two give an idea of the "spread" of the energies found.

If no new minimum is found over `SWARMMINITER` iterations, the algorithm is considered to be converged and the best solutions are taken to the final optimization step.

4.13.7 The final steps

After the evolution step, a total of $max(sqrt(SwarmPopSize), 5)$ structures are taken from the set of solutions for a final full optimization. The output looks like:

```
Running final optimization
   Maximum number of structures      7
   Minimum energy difference         0.10 kcal/mol
   Maximum RMSD                      0.25 Angs
   Optimization strategy             regular
   Coordinate system                 redundant 2022
   Fixed host                        false
```

To avoid redundant solutions being optimized here, a check is made such that only structures with an energy difference of at least a `Minimum energy difference` and `Maximum RMSD` (obtained after an optimal rotation using the quaternion approach) are taken. The coordinate system is also be automatically set to Cartesian if there are more than 300 atoms in total. Set `AUTOCOORDSYS FALSE` to avoid that or `%GEOM COORDSYS CARTESIAN END` to use Cartesian coordinates all the way during the docking process.

If some optimizations fail during the process, they will be flagged as `optimization failed`, as in the example below:

```
   Struc    Eopt        Interaction Energy        Time
            (Eh)                (kcal/mol)       (min)
   ----------------------------------------------

    1       -10.149006          -4.968378        0.01
    2       -10.149005          -4.967965        0.01
    3        optimization failed                 0.01
    4       -10.149007          -4.968641        0.01
   (...)
```

That only means some of the solutions failed to fully optimize at the end, and on the printed file their energy is set to 1000 Eh to show the job was not completed. If you want to increase the number of iterations, or change the final convergence criteria, just change them using the `%GEOM` block as usual.

4.13.8 Adding a bond bias to the docking process

The bond biases available for geometry optimization (see Section *Bond Bias Potentials*) can also be applied to the DOCKER, but the input here is slightly different.

As the DOCKER does not know the total number of atoms from the start, the `%DOCKER` block needs the first atom numbers from `GUEST` and `HOST` in that order and not the number of atoms from the final complex.

For example, to add a bond bias between atom 2 from the `GUEST` and atom 19 from the `HOST` during the docking process, please use:

```
%DOCKER
  BIAS { B 2 19 } END
END
```

The same parameters and their defaults from the `%GEOM` block are used here.

> **Note**
>
> This will also be applied to the SOLVATOR if the `CLUSTERMODE DOCKING` (default) is used. The `GUEST` atom number corresponds to that of the solvent, as printed in the output.

Important

If biases are added from the %DOCKER block, they will override any other bias from the %GEOM block.

4.13.9 Defining the center and extent of the grid

As explained above, the placement of GUEST molecules is done with the help of a spatial grid. By default, the grid is built around the centroid of the host molecule. However, in cases a different position for the grid is wanted, it can be changed by setting an arbitrary center and extents.

Its center can be defined in terms of Cartesian coordinates simple via:

```
%DOCKER
  GRIDCENTER 1.00, 0.642, -1.234 # coordinates in Angstroem
END
```

or a list of atoms can also be given, and the centroid of those atoms will be taken. If a single atom is given, that will of course be the center:

```
%DOCKER
  GRIDCENTERATOMS {0:5 7} END # numbers starting from 0, as usual
END
```

Important

If GRIDCENTERATOMS were given and the HOST is optimized, which is the default case, the grid center will be defined only **after** the geometry is optimized. For the GRIDCENTER option, the center will be fixed at the desired Cartesian coordinates.

The extent of the grid around that center can be controlled by:

```
%DOCKER
  GRIDEXTENT 15 # extent in Angstroem
END
```

and all x, y and z coordinates will expand that much from the center. It has to be a cube, and all dimensions will be the same. If an extent is not given, the default values will be used.

Note

There is no problem setting the grid extent to regions where atoms are present. The grid will be only built around these, same as for the default case.

4.13.10 General Tips

1. In order to quick change the PES of the docking process, you can use !DOCK(GFN2-XTB), !DOCK(GFN1-XTB), !DOCK(GFN0-XTB) or !DOCK(GFN-FF).
2. For a quicker and less accurate docking the input !QUICKDOCK is also available.
3. There are four levels of "docking" procedure, from simple to more elaborate: SCREENING, QUICK, NORMAL (default) and COMPLETE, which can be given as %DOCK DOCKLEVEL COMPLETE END.
4. To increase the final number of structures optimized in the end, just change NOPT under %DOCKER.
5. If the guest topology is somehow changed during the docking process, that structure will be discarded. This can be turned off by setting %DOCK CHECKGUESTTOPO FALSE END.

Important

By the time of the ORCA6.1 release, this algorithm was still not published. Publications are under preparation.

4.13.11 Keywords

A collection of GOAT related simple input keywords is given in Table 4.11. All `%docker` block options are summarized in Table 4.12.

Table 4.11: Simple input keywords for the DOCKER algorithm.

Keyword	Description
`QUICKDOCK`	Set `DOCKLEVEL QUICK`
`NORMALDOCK`	Set `DOCKLEVEL NORMAL`
`COMPLETEDOCK`	Set `DOCKLEVEL COMPLETE`
`DOCK(GFN-FF)`	Set `EVPES GFNFF`
`DOCK(GFN0-XTB)`	Set `EVPES GFN0XTB`
`DOCK(GFN1-XTB)`	Set `EVPES GFN1XTB`
`DOCK(GFN2-XTB)`	Set `EVPES GFN2XTB`

Table 4.12: `%docker` block input keywords for t

Keyword	Options	Description
`GUEST`	`"filename.xyz"`	An .xyz file (can be multistructure), from where the guest(s) will be read. Can contain different charges a
`DOCKLEVEL`	`screening`	Defines a general strategy for docking.
	`quick`	Will alter things like that population density and final number of optimized structrures. default is `normal`
	`normal`	
	`complete`	
`PRINTLEVEL`	`low`	
	`normal`	Default
	`high`	Will print many extra files!
`NREPEATGUEST`	`1`	Number of times to repeat the content of the "GUEST" file
`CUMULATIVE`	`true`	Add the contents of the "GUEST" file one on top of each other? default is `false`, meaning each will be
`GUESTCHARGE`	`0`	Can be used to defined a charge for the guest (default 0)
`GUESTMULT`	`3`	Same for multiplicty (default 1) both can also be given via the comment line of the "filename.xyz", see ab
`RAMDOMSEED`	`true`	Whether to allow for the process to be trully random and will give different results everytime (default `fal`
`FIXHOST`	`true`	Freeze coordinatef for the HOST during all steps? (default `false`)
`OPTLEVEL`	`sloppyopt`	Use default optimization criteria inside the DOCKER
	`looseopt`	
	`normalopt`	
`AUTOCOORDSYS`	`false`	Automatically use Cartesian coordinates for the docking process (default `false`)
`ALLOWMETALCOORD`	`true`	will allow the DOCKER to place atoms close enough to metals such as to create coordination bonds. by d
`MBONDFAC`	`1.0`	A multiplying factor for the radii when defining a bond with metals. Default is 1.0
Grid options:		
`GRIDCENTER`	`1.00, 0.642, -1.234`	Cartesian coordinates for the center in Angström
`GRIDCENTERATOMS`	`{0:5 7} end`	A list of atoms from which the centroid will be taken
`GRIDEXTENT`	`15`	The extent, from the center, in one dimension in Angström
Swarm intelligence step:		
`POPDENSITY`	`5.0`	Population density per Angström squared. Default depends on the Docking level
`POPSIZE`	`500`	A fixed number for the population size
`SWARMMAXITER`	`10`	Maximum number of iterations
`SWARMMINITER`	`3`	Minimum number to check for convergence and minimum required of equal steps to signal convergence
`SWARMPOPDENSITY`	`0.5`	The population density based on the HOST grid
`SWARMPOPSIZE`	`150`	A fixed number for the swarm population size
`SWARMPES`	`cdp`	Which PES to use **only** during the evolution step. The default `cdp` is **a lot** faster than the others.
	`gfnff`	Can be different from the final optimization.
	`gfn0xtb`	
	`gfn1xtb`	
	`gfn2xtb`	
`CHECKTOPO`	`false`	Check the topology of both host and guest during the docking process? Default is true.
Pre optimizations		
`PREOPT`	`false`	Controls whether the PreOpt step will be done. Default `true`.
Final optimizations:		
`NOPT`	`10`	A fixed number of structures to be optimized
`NOOPT`	`false`	Do not optimize any structure at all? (default `false`)

CHAPTER

FIVE

SPECTROSCOPY AND PROPERTIES

5.1 Population Analysis

Atomic population related quantities are not real molecular properties since they are not observables. They are nevertheless highly useful for interpreting experimental and computational findings. Various ways of analyzing a computed SCF wavefunction are available within ORCA. By default, ORCA provides very detailed information about calculated molecular orbitals and bonds through *Mulliken*, *Löwdin*, and *Mayer* population analyses.

See also

As these methods typically create a very large amount of specific output, we generally recommend to read the *Control of Output* section.

The `!ReducedPOP` keyword is of particular use as it reduces the printed information in the population analysis section, providing orbital population of each atom with percent contribution per basis function type. This is highly useful in figuring out the character of the MOs.

5.1.1 Mulliken Population Analysis

The Mulliken population analysis [574] is, despite all its known considerable weaknesses, the standard in most quantum chemical programs. It partitions the total density using the assignment of basis functions to given atoms in the molecules and the basis function overlap. If the total charge density is written as $\rho(\vec{r})$ and the total number of electrons is N we have:

$$\int \rho(\vec{r})\, d\vec{r} = N \tag{5.1}$$

and from the density matrix **P** and the basis functions ϕ:

$$\rho(\vec{r}) = \sum_{\mu\nu} P_{\mu\nu} \phi_\mu(\vec{r})\, \phi_\nu(\vec{r}) \tag{5.2}$$

therefore:

$$\int \rho(\vec{r})\, d\vec{r} = \sum_{\mu\nu} P_{\mu\nu} \underbrace{\int \phi_\mu(\vec{r})\, \phi_\nu(\vec{r})\, d\vec{r}}_{S_{\mu\nu}} \tag{5.3}$$

$$= \sum_{\mu\nu} P_{\mu\nu} S_{\mu\nu} \tag{5.4}$$

After assigning each basis function to a given center this can be rewritten:

$$= \sum_A \sum_B \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.5}$$

$$= \sum_A \sum_\mu {}^A \sum_\nu {}^A P_{\mu\nu}^{AA} S_{\mu\nu}^{AA} + 2 \sum_A \sum_{B<A} \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.6}$$

Mulliken proposed to divide the second term equally between each pair of atoms involved and define the number of electrons on center A, N_A, as:

$$N_A = \sum_\mu {}^A \sum_\nu {}^A P_{\mu\nu}^{AA} S_{\mu\nu}^{AA} + \sum_{B \neq A} \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.7}$$

such that $\sum_A N_A = N$. The charge of an atom in the molecule is then:

$$Q_A = Z_A - N_A \tag{5.8}$$

where Z_A is the core charge of atom A. The cross terms between pairs of basis functions centered on different atoms is the overlap charge and is used in ORCA to define the Mulliken bond order:

$$B_{AB} = 2 \sum_\mu {}^A \sum_\nu {}^B P_{\mu\nu}^{AB} S_{\mu\nu}^{AB} \tag{5.9}$$

Basic Usage

The Mulliken population analysis can be invoked by the `!MULLIKEN` input keyword:

```
! MULLIKEN
```

It can further be requested via the `Print[ P_Mulliken ] 1` keyword in the `%output` block

```
%output
  Print[ P_Mulliken ] 1  # default = on
end
```

A number of additional options can be specified in the `%output` block to control the details of the Mulliken population analysis. By default the Mulliken population analysis is turned on.

```
%output
  Print[ P_AtCharges_M  ] 1     # Print atomic charges
  Print[ P_OrbCharges_M ] 1     # Print orbital charges
  Print[ P_FragCharges_M] 1     # Print fragment charges
  Print[ P_BondOrder_M  ] 1     # Print bond orders
  Print[ P_FragBondOrder_M ] 1  # Print fragment b.o.
  Print[ P_ReducedOrbPop_M ] 1  # Print reduced orb. Charges
  Print[ P_AtPopMO_M    ] 1     # Print atomic charges in each MO
  Print[ P_OrbPopMO_M   ] 1     # Print orbital populaiton for each MO
  Print[ P_ReducedOrbPopMO_M] 1 # Print reduced orbital pop for each MO
  Print[ P_FragPopMO_M  ] 1     # Print the fragment population for for each MO
end
```

These options allow to get very detailed information about the computed wavefunctions and is much more convenient than to look at the MOs directly. A "reduced orbital population" is a population per angular momentum type. For example the sum of populations of each p_z orbital at a given atom is the reduced orbital population of the p_z function.

Note that for *finite temperature HF or KS-DFT* calculations (`SmearTemp` > 0 K, fractional occupation numbers or *FOD analysis*), only the Mulliken reduced orbital charges based on ρ^{FOD} will be printed.

5.1.2 Löwdin Population Analysis

The Löwdin analysis [180] is somewhat more straightforward than the Mulliken analysis. In the Löwdin method one changes to a basis where all overlap integrals vanish. This is accomplished via Löwdins symmetric orthogonalization matrix $\mathbf{S}^{-1/2}$. Using this transformation matrix the new basis functions are multicentered but are in a least square sense as similar as possible to the original, strictly localized, atomic basis functions. The similarity of the transformed functions and original functions is explored in the population analysis. The density matrix transforms as:

$$\mathbf{P}^L = \mathbf{S}^{1/2}\mathbf{P}\mathbf{S}^{1/2} \tag{5.10}$$

Then the atomic populations are:

$$N_A = \sum_{\mu}{}^{A} P^L_{\mu\mu} \tag{5.11}$$

The bond order is defined from the Wiberg index [575] that was first used in the context of semiempirical methods (that are formulated in the Löwdin basis right from the start):

$$B_{AB} = \sum_{\mu}{}^{A} \sum_{\nu}{}^{B} \left(P^L_{\mu\nu}\right)^2 \tag{5.12}$$

Basic Usage

The Löwdin population analysis can be invoked by the `!LOEWDIN` input keyword. By default the Löwdin population analysis is turned on and provides some more detail than the Mulliken analysis.

```
! MULLIKEN
```

It can further be requested via the `Print[ P_Loewdin ] 1` keyword in the `%output` block:

```
%output
  Print[ P_Loewdin ] 1  # default = on
end
```

The details of the Löwdin population analysis printout can be controlled via the `%output` block:

```
%output
  Print[ P_AtCharges_L  ] 1     # Print atomic charges
  Print[ P_OrbCharges_L ] 1     # Print orbital charges
  Print[ P_FragCharges_L] 1     # Print fragment charges
  Print[ P_BondOrder_L  ] 1     # Print bond orders
  Print[ P_FragBondOrder_L ] 1  # Print fragment b.o.
  Print[ P_ReducedOrbPop_L ] 1  # Print reduced orb. Charges
  Print[ P_AtPopMO_L    ] 1     # Print atomic charges in each MO
  Print[ P_OrbPopMO_L   ] 1     # Print orbital population for each MO
  Print[ P_ReducedOrbPopMO_L] 1 # Print reduced orbital pop for each MO
  Print[ P_FragPopMO_L  ] 1     # Print the fragment population for each MO
end
```

In addition one can set the threshold for the printing of the bond order in the `%method` block.

```
%method
  LOEWDIN_BONDORDERTHRESH 0.05
end
```

5.1.3 Frontier Molecular Orbital Populations

As the frontier orbitals are typically of specific interest, *Mulliken* and *Löwdin* populations of the HOMO and LUMO can be requested via the "`FMOPop`" keyword:

```
! FMOPop
```

The respective Mulliken and Loewdin population of the HOMO and LUMO frontier orbitals will be printed in the output:

```
---------------------------------------------
FRONTIER MOLECULAR ORBITAL POPULATION ANALYSIS
---------------------------------------------

ANALYZING ORBITALS: HOMO=   6 LUMO=   7

----------------------------------------------------------------------
   Atom        Q(Mulliken)     Q(Loewdin)     Q(Mulliken)     Q(Loewdin)
              <<<<<<<<<<<<HOMO>>>>>>>>>>>>   <<<<<<<<<<<<LUMO>>>>>>>>>>>>
----------------------------------------------------------------------
   0-C         0.937186        0.906827       0.804044        0.755610
   1-O         0.062814        0.093173       0.195956        0.244390
----------------------------------------------------------------------
```

5.1.4 Mayer Population Analysis

Mayers bonding analysis [576, 577, 578, 579] is another creative attempt to define chemically useful indices. The Mayer atomic charge is identical to the Mulliken charge. The Mayer bond order is defined as:

$$B_{AB} = \sum_{\mu}^{A} \sum_{\nu}^{B} (\mathbf{PS})_{\mu\nu} (\mathbf{PS})_{\nu\mu} + (\mathbf{RS})_{\mu\nu} (\mathbf{RS})_{\nu\mu} \tag{5.13}$$

Here **P** is the total electron density matrix and **R** is the spin-density matrix. These Mayer bond orders are very useful. Mayer's total valence for atom A is defined as:

$$V_A = 2N_A - \sum_{\mu}^{A} \sum_{\nu}^{A} (\mathbf{PS})_{\mu\nu} (\mathbf{PS})_{\nu\mu} \tag{5.14}$$

In normal bonding situations and with normal basis sets V_A should be reasonably close to the valence of atom A in a chemical sense (i.e. close to four for a carbon atom). The bonded valence is given by:

$$X_A = V_A - \sum_{B \neq A} B_{AB} \tag{5.15}$$

and finally the free valence (a measure of the ability to form further bonds) is given by:

$$F_A = V_A - X_A \tag{5.16}$$

Basic Usage

The Mayer population analysis can be invoked via the `!MAYER` keyword:

```
! MAYER
```

It can further be requested via the `Print[ P_Mayer ] 1` keyword in the `%output` block:

```
%output
  Print[ P_Mayer ] 1  # default = on
end
```

The output is rather simple and short and can not be further controlled. By default the Mayer population analysis is turned on. In addition one can set the threshold for the printing of the bond order in the `%method` block.

```
%method
  MAYER_BONDORDERTHRESH 0.1
end
```

5.1.5 Natural Population Analysis

If you have access to a version of the `gennbo` program from Weinhold's group[1] you can request the Natural Population analysis via the *NBO interface*.

Important

The NPA is only performed if the NBO program is provided properly. If not, ORCA will skip the NPA analysis.

Basic Usage

The Natural population analysis can be invoked via the `!NPA` keyword:

```
! NPA
```

It can further be requested via the `Print[ P_NPA ] 1` keyword in the `%output` block:

```
%output
  Print[ P_NPA ] 1  # default = off
end
```

5.1.6 Hirshfeld Population Analysis

The partitioning method by Hirshfeld is one of the most used approaches in the so-called atoms in molecules (AIM) methods.[580] In this case, the AIM density of atom A, $\rho_A(\vec{r})$ is written as:

$$\rho_A(\vec{r}) = \rho(\vec{r}) w_A(\vec{r}) \tag{5.17}$$

Here, $\rho(\vec{r})$ is the total charge density at position $\vec{r}$, and $w_A(\vec{r})$ a weighting function, that within the Hirshfeld method is equal to:

$$w_A(\vec{r}) = \frac{\rho_A^0(\vec{r})}{\rho^0(\vec{r})} \tag{5.18}$$

where $\rho_A^0(\vec{r})$ is the pro-atomic density of atom A and $\rho^0(\vec{r}) = \sum_A \rho_A^0(\vec{r})$ the pro-molecular density. The ratio in eq. (5.17) is known as *stockholder*. From eqs. (5.17) and (5.18) one can calculate the Hirshfeld charges as:

$$Q_A^{\text{Hirsh.}} = Z_A - \int \rho_A(\vec{r}) d\vec{r} \tag{5.19}$$

In ORCA, the pro-atomic density within the Hirshfeld method is calculated via density fitting with a set of Gaussian s-functions per element.

[1] Information about the NBO program can be found at http://nbo7.chem.wisc.edu

Basic Usage

The Hirshfeld population analysis can be invoked by the `!HIRSHFELD` input keyword.

```
! HIRSHFELD
```

It can further be requested via the `Print[ P_Hirshfeld ] 1` keyword in the `%output` block:

```
%output
  Print[ P_Hirshfeld ] 1  # default = off
end
```

Example

As an example we request the Hirshfeld charges for a water molecule:

```
!HF cc-pVDZ TightSCF HIRSHFELD

* xyz 0 1
  O    0.00000006589375       0.00157184228646       0.00000000004493
  H    0.77316868532439      -0.58666889665624      -0.00000000000005
  H   -0.77316876182122      -0.58666895650640      -0.00000000000005
*
```

ORCA prints the following information in the output file:

```
------------------
HIRSHFELD ANALYSIS
------------------

Total integrated alpha density =      4.999998580
Total integrated beta density  =      4.999998580

  ATOM     CHARGE      SPIN
   0 O   -0.333756    0.000000
   1 H    0.166879    0.000000
   2 H    0.166879    0.000000

  TOTAL   0.000003    0.000000
```

5.1.7 MBIS Charges

The Minimal Basis Iterative Stockholder (MBIS) method is a variant of the Hirshfeld method.[581] The idea behind this approach is that the pro-atomic density $\rho_A^0(\vec{r})$ is expanded in a minimal set of atom-centered s-type Slater functions $\rho_{Ai}^0(\vec{r})$:

$$\rho_A^0(\vec{r}) = \sum_{i=1}^{m_A} \rho_{Ai}^0(\vec{r}) \tag{5.20}$$

with $\rho_{Ai}^0(\vec{r})$ equal to:

$$\rho_{Ai}^0(\vec{r}) = \frac{N_{Ai}}{\sigma_{Ai}^3 8\pi} \exp\left(-\frac{\left|\vec{r} - \vec{R_A}\right|}{\sigma_{Ai}}\right) \tag{5.21}$$

Here, m_A is the number of shells of atom A. The populations N_{Ai}, and the widths σ_{Ai} can be written as:

$$N_{Ai} = \int \rho(\vec{r}) \frac{\rho_{Ai}^0(\vec{r})}{\rho^0(\vec{r})} d\vec{r} \tag{5.22}$$

$$\sigma_{Ai} = \frac{1}{3N_{Ai}} \int \rho(\vec{r}) \frac{\rho^0_{Ai}(\vec{r})}{\rho^0(\vec{r})} \left|\vec{r} - \vec{R_A}\right| d\vec{r} \tag{5.23}$$

In order to compute the AIM densities $\rho_A(\vec{r})$, the MBIS method uses an iterative algorithm where: (1) an initial guess is generated for the set of N_{Ai} and σ_{Ai} and the pro-atomic densities are calculated through eqs. (5.20) and (5.21), (2) the new set of N_{Ai} and σ_{Ai} are obtained via eqs. (5.22) and (5.23), (3) if convergence is reached for $\rho_A(\vec{r})$, the iterative process stops, otherwise we go back to (1) but now one uses the last estimates for N_{Ai} and σ_{Ai}.

Once, the MBIS iterative process stops, the MBIS charges are calculated as:

$$Q_A^{\text{MBIS.}} = Z_A - \int \rho_A(\vec{r}) d\vec{r} \tag{5.24}$$

Basic Usage

The MBIS population analysis can be invoked by the `!MBIS` input keyword.

```
! MBIS
```

It can further be requested via the `Print[ P_MBIS ] 1` keyword in the `%output` block:

```
%output
  Print[ P_MBIS ] 1  # default = off
end
```

The convergence threshold for the MBIS charges is set to 10^{-6}. However, it can be changed via the tag `MBIS_CHARGETHRESH` in the `%method` block:

```
%method
  MBIS_CHARGETHRESH 0.0001
end
```

MBIS Quantities

ORCA can also print the following MBIS-related quantities:

1. atomic dipole moments,
2. atomic quadrupole moments,
3. atomic octupole moments, and
4. third radial moment of the MBIS density $\left(\langle r^3\rangle_A = \int \left|\vec{r} - \vec{R_A}\right|^3 \rho_A(\vec{r}) d\vec{r}\right)$.

The printing of these properties is controlled by the `MBIS_LARGEPRINT` in the `%method` block:

```
%method
  MBIS_LARGEPRINT true  # default = false
end
```

If this option is activated, an extra iteration is performed after reaching the convergence threshold for the charges.

The origin for the calculation of the atomic dipole, quadrupole and octupole moments is the center of each atom (default). The user can also define a global origin (independent of the atom) through the `MBIS_ORIGIN_MULT` keyword in the `%method` block:

```
%method
  MBIS_ORIGIN_MULT  CenterOfCoords     # origin of coordinate system (0,0,0)
                    CenterOfMass       # center of mass
                    CenterOfNucCharge  # center of nuclear charge
                    CenterXYZ          # arbitrary position, set coordinates with␣
```

(continues on next page)

(continued from previous page)

```
↪MBIS_ORIMULT_XYZ
                    CenterOfEachAtom   # center of each atom (default)

  MBIS_ORIMULT_XYZ  x,y,z # set the coordinates, otherwise 0,0,0 (unit: Angstrom)
end
```

Example

If we request the MBIS charges for a HF calculation at the cc-pVDZ level of a chloroform molecule:

```
!HF cc-pVDZ TightSCF MBIS

* xyz 0 1
  C   -0.00000997794639     -0.00091664148112      0.45499807439812
  H    0.00000069467312      0.00031189002174      1.53703126401237
  Cl   0.00003188789531      1.69433732001280     -0.08420513240263
  Cl   1.46635420502892     -0.84684178730039     -0.08421103795485
  Cl  -1.46637680965097     -0.84689178125304     -0.08420916805301
*
```

ORCA prints the following information at the end of the output file:

```
------------------
MBIS ANALYSIS
------------------

Convergence threshold (charges)      ...        1.000e-06
Number of iterations                 ...        46

Total integrated alpha density       ...     29.000001385
Total integrated beta density        ...     29.000001385

  ATOM     CHARGE    POPULATION     SPIN
   0 C    0.208633    5.791367    0.000000
   1 H    0.169417    0.830583    0.000000
   2 Cl  -0.126877   17.126877    0.000000
   3 Cl  -0.125586   17.125586    0.000000
   4 Cl  -0.125590   17.125590    0.000000

  TOTAL  -0.000003   58.000003    0.000000

 MBIS VALENCE-SHELL DATA:
  ATOM   POPULATION   WIDTH(A.U.)
   0 C    4.122213    0.508675
   1 H    0.830583    0.358785
   2 Cl   8.532439    0.524031
   3 Cl   8.531380    0.523959
   4 Cl   8.531381    0.523959
```

The second block corresponds to the valence Slater function, which is caracterized by its population $N_{A,v}$ and width $\sigma_{A,v}$.

5.1.8 CHELPG Charges

Atomic charges can also be calculated using the CHarges from ELectrostatic Potentials using a Grid-based (CHELPG) method according to Breneman and Wiberg.[582] In this approach, the atomic charges are fitted to reproduce the electrostatic potential on a regular grid around the molecule, while constraining the sum of all atomic charges to the molecule's total charge. An additional constraint can be added, so the CHELPG charges also reproduce the total dipole moment of the molecule.

Basic Usage

In ORCA the CHELPG charges can be (i) requested within a calculation, or (ii) calculated via the standalone *orca_chelpg* utility. To follow path (i) one has to add the `!CHELPG` simple input keyword to the input file.

```
! CHELPG
```

Further settings can be controlled via the `%chelpg` block.

```
%chelpg
  GRID 0.3       # Spacing of the regular grid in Angstroems
  RMAX 2.8       # Maximum distance of all atoms to any gridpoint in Angstroems
  VDWRADII COSMO # VDW Radii of Atoms, COSMO default
           BW    # Breneman, Wiberg radii
  DIPOLE FALSE   # If true, then the charges also reproduce the total dipole moment
end
```

By default the program uses the COSMO VDW radii for the exclusion of gridpoints near the nuclei, as these are defined for all atoms. The BW radii are similar, but only defined for very few atom types.

The charges may exhibit some dependence on the molecule's orientation in space, or some artificial variations in symmetric molecules. These effects can be minimized by increasing the CHELPG grid size, either by setting the `GRID` parameter in the `%chelpg` block, or by using the `!CHELPG(LARGE)` simple input keyword.

```
! CHELPG(LARGE)
```

If one wants that the calculated CHELPG charges reproduce the total dipole moment of the molecule, as well as the electrostatic potential, then the following tag has to be added to the `%chelpg` block:

```
%chelpg
  DIPOLE TRUE  # The default is set to FALSE
end
```

In particular, the constraint affects the x,y,z components of the total dipole moment, so they reproduce the exact three components of the total dipole moment calculated via one-electron integrals.

5.1.9 RESP Charges

ORCA also features the possibility of calculating the so-called RESP charges [583], which are essentially CHELPG charges with some restraint to avoid over fitting. These can be invoked with the defaults by simply using `!RESP` on the main input.

These RESP charges can have two types of penalty function: quadratic or hyperbolic (default), as described on the original paper. These require up to two penalty values, which are by default given by $a = 0.0005$ and $b = 0.1$.

These values can be changed inside the `%CHELPG` block by using:

```
%chelpg
  RESPA 0.01   # for the a value
  RESPB 0.002  # for the b value
end
```

and the penalty function can be chosen with:

```
%chelpg
  RESPPENALTY QUADRATIC # default is HYPREBOLIC
end
```

Also, if needed, the individual values for the a parameters and reference charges q_0 can be given via a special file with:

```
%chelpg
  RESPWFILE     "wfilename"   # these will define the weights (or a parameters)␣
↪for each atom
  RESPQREFFILE  "qfilename"   # these might define reference charges different␣
↪from zero
end
```

The format for both these files is similar to a `.xyz` file, where the first line has the number of atoms, the second line is just a comment, and every subsequent line contains a single number which is either the weights or the reference charges per atom. They can not be given on a single file, there must be two different ones.

> **Note**
>
> The results from the RESP files obtained here are **not** necessarily the same one would get from other software. Two main reasons are: the grids are not the same and some other software use a multilevel approach, first starting with a certain value for the weights and then switching to a different value, which is not the case here.

5.1.10 Local Spin Analysis

It is common practice in chemistry to think about the interaction of open-shell systems in terms of local spin states. For example, in dimeric or oligomeric transition metal clusters, the 'exchange coupling' between open shell ions that exist locally in high-spin states is extensively studied. Diradicals would be typical systems in organic chemistry that show this phenomenon. In quantum mechanics, however, the total spin is not a local property, but instead a property of the system as a whole. The total spin squared, S^2, and its projection onto the z-axis, S_z, commute with the non-relativistic Hamiltonian and hence, the eigenfunctions of the non-relativistic Hamiltonian can be classified according to good quantum numbers S and M according to:

$$\mathbf{S}^2 \left|\Psi^{SM}\right\rangle = S(S+1) \left|\Psi^{SM}\right\rangle$$

$$S_z \left|\Psi^{SM}\right\rangle = M \left|\Psi^{SM}\right\rangle$$

where $\left|\Psi^{SM}\right\rangle$ is an exact eigenfunction of the non-relativistic Hamiltonian or an approximation to it that conserves the total spin as a good quantum number. The total spin itself is given by the sum over the individual electron spins as:

$$\mathbf{S} = \sum_i \mathbf{s}(i)$$

And hence,

$$\mathbf{S}^2 = \sum\nolimits_{i,j} \mathbf{s}(i)\mathbf{s}(j)$$

is a two-electron property of the system. It is obviously not trivial to relate the chemically very meaningful concept of local spin to a rigorous quantum mechanical treatment. While there are various proposals of how to deal with this problem, we follow here a proposal of Clark and Davidson[584]. The following equations are implemented in the SCF and CASSCF modules of Orca.

Clark and Davidson define fragment projection operators with the property:

$$P_A P_B = \delta_{AB} P_A$$

and:

$$\sum_A P_A = 1$$

Then using this identity:

$$\mathbf{S} = \sum_i \sum_A \mathbf{s}(i) P_{A(i)}$$

$$\mathbf{S} = \sum_A \sum_i \mathbf{s}(i) P_A(i)$$

$$= \sum_A \mathbf{S}_A(i)$$

they show that the local spin operators obey the standard relations for spin operators:

$$\mathbf{S}_A = \mathbf{S}_A^\dagger$$

$$\mathbf{S}_A \times \mathbf{S}_A = i\hbar \mathbf{S}_A$$

Hence

$$\mathbf{S}^2 = \sum_A \sum_B \mathbf{S}_A \mathbf{S}_B$$

But then importantly:

$$\mathbf{S}_A \mathbf{S}_B = \sum_i \sum_j \mathbf{s}(i)\mathbf{s}(j) P_A(i) P_B(j)$$

$$= \tfrac{3}{4}\delta_{AB} \sum_A P_A(i) + \sum_i \sum_{j>i} \mathbf{s}(i)\mathbf{s}(j)\{P_A(i)P_B(j) + P_A(j)P_B(i)\}$$

With the first- and second-order density matrix:

$$\gamma(\mathbf{x}, \mathbf{x}') = N \int \Psi(\mathbf{x}, \mathbf{x}_2, ..., \mathbf{x}_N) \Psi^*(\mathbf{x}', \mathbf{x}_2, ..., \mathbf{x}_N) d\mathbf{x}_2 ... d\mathbf{x}_N$$

$$\Gamma(\mathbf{x}_1, \mathbf{x}'_1; \mathbf{x}_2, \mathbf{x}'_2) = \binom{N}{2} \int \Psi(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) \Psi^*(\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}_N) d\mathbf{x}_3 ... d\mathbf{x}_N$$

(with $\binom{N}{2} = \frac{1}{2}N(N-1)$). Then:

$$\langle \mathbf{S}_A \mathbf{S}_B \rangle = \tfrac{3}{4}\delta_{AB} tr(\gamma P_A) + 2tr(P_A(1)P_B(2)\mathbf{s}(1)\mathbf{s}(2)\Gamma(1,1;2,2))$$

In terms of the number of electrons on site 'A'and the expectation value of S_z^A

$$\left\langle S_z^A \right\rangle = \tfrac{1}{2} tr(\gamma^{\alpha-\beta} P_A)$$

$$\left\langle N^A \right\rangle = tr(\gamma^{\alpha+\beta} P_A)$$

in terms of molecular orbitals:

$$\left\langle S_z^A \right\rangle = \tfrac{1}{2} \sum_{p,q} \gamma_{pq}^{\alpha-\beta} \langle p | P_A | q \rangle$$

$$\left\langle N^A \right\rangle = \sum_{p,q} \gamma_{pq}^{\alpha+\beta} \langle p | P_A | q \rangle$$

McWeeny and Kutzelnigg[585] show that for the expectation value of s(1)s(2), the relevant irreducible part of the two-body density can be expressed in terms of the spinless density matrix of second order:

$$R_0^{(0)}(1,1';2,2') = -\tfrac{1}{3}\Gamma(1,1';2,2') - \tfrac{2}{3}\Gamma(2,1';1,2')$$

$$= -\tfrac{1}{3}\sum_{pqrs}\Gamma^{pq}_{rs}p(1)q(1')r(2)s(2') + 2\Gamma^{pq}_{rs}p(2)q(1')r(1)s(2')$$

$$= -\tfrac{1}{3}\sum_{pqrs}(\Gamma^{pq}_{rs} + 2\Gamma^{rq}_{ps})p(1)q(1')r(2)s(2')$$

with a normalization factor of $\frac{3}{4}$ after spin integration. Hence using this:

$$\langle \mathbf{S}_A\mathbf{S}_B\rangle = \tfrac{3}{4}\delta_{AB}tr(\gamma P_A) + \tfrac{6}{4}tr(P_A(1)P_B(2)R_0^{(0)}(1,1;2,2))$$

And then performing the integral:

$$\langle \mathbf{S}_A\mathbf{S}_B\rangle = \tfrac{3}{4}\delta_{AB}tr(\gamma P_A) - \underbrace{\tfrac{6}{4}\tfrac{1}{3}}_{\frac{1}{2}}\sum_{pqrs}(\Gamma^{pq}_{rs} + 2\Gamma^{rq}_{ps})P^A_{pq}P^B_{rs}$$

This is the final and perhaps most compact equation. The projection operator can be defined in very many different ways. The easiest is to Löwdin orthogonalize the basis set:

$$\left|\mu^A_L\right\rangle = \sum_{\nu^A}\left|\nu^A\right\rangle S^{-1/2}_{\mu\nu}$$

where L denotes the Löwdin basis. This means that molecular orbitals are expressed in the orthogonal basis as:

$$\mathbf{c}_L = \mathbf{S}^{+1/2}\mathbf{c}$$

and the density as:

$$\mathbf{P}_L = \mathbf{S}^{+1/2}\mathbf{P}\mathbf{S}^{+1/2}$$

The fragment projector is defined as:

$$P_A = \sum_{\mu_L\in A}\left|\mu_L\right\rangle\left\langle\mu_L\right|$$

Clark and Davidson suggest a slightly more elaborate projector in which first, the intra-fragment overlap is eliminated. This happens with a matrix U that for two fragments takes form:

$$\mathbf{U} = \begin{pmatrix}\mathbf{S}_A^{-1/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_B^{-1/2}\end{pmatrix}$$

where is the block of basis functions belonging to fragment A. Likewise:

$$\mathbf{U}^{-1} = \begin{pmatrix}\mathbf{S}_A^{+1/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}_B^{+1/2}\end{pmatrix}$$

Then the 'pre-overlap'is:

$$\tilde{\mathbf{S}} = \mathbf{U}^\dagger\mathbf{S}\mathbf{U}$$

This contains the unit matrix in the intra-fragment blocks and non-zero elements elsewhere. This overlap matrix is the finally orthogonalized to obtain the globally orthogonal Löwdin basis. We finally transform the MO coefficients by the following transformation:

$$\mathbf{c}_L = \mathbf{S}^{+1/2}\mathbf{U}^{-1}\mathbf{c}$$

For the projectors, operating with the two MOs i and j gives:

$$\langle i|P_A|j\rangle = \sum_{\mu_L\in A}\sum_{\kappa^B_L\tau^C_L}\left\langle\kappa^B_L|\mu^A_L\right\rangle\left\langle\mu^A_L|\tau^C_L\right\rangle c^L_{\kappa i}c^L_{\tau j}$$

$$= \sum_{\mu_L \in A} \sum_{\kappa_L^B \tau_L^C} \delta_{AB} \delta_{AC} \delta_{\kappa\mu} \delta_{\tau\mu} c_{\kappa i}^L c_{\tau j}^L$$

$$= \sum_{\mu_L \in A} c_{\mu i}^L c_{\mu j}^L$$

Herrmann et al.[586] give the correct expression of the expectation values for a single spin-unrestricted determinant

$$\begin{aligned}
\langle \mathbf{S}_A \mathbf{S}_B \rangle &= \tfrac{3}{4} \delta_{AB} \left\{ \sum_i P_{ii}^A + \sum_{\bar{i}} P_{\bar{i}\bar{i}}^A \right\} \\
&+ \tfrac{1}{4} \left\{ \sum_{ij} P_{ii}^A P_{jj}^B + \sum_{\bar{i}\bar{j}} P_{\bar{i}\bar{i}}^A P_{\bar{j}\bar{j}}^B - \sum_{ij} P_{ij}^A P_{ij}^B - \sum_{\bar{i}\bar{j}} P_{\bar{i}\bar{j}}^A P_{\bar{i}\bar{j}}^B - \sum_{\bar{i}j} P_{\bar{i}\bar{i}}^A P_{jj}^B - \sum_{i\bar{j}} P_{ii}^A P_{\bar{j}\bar{j}}^B \right\} \\
&- \sum_{i\bar{j}} P_{i\bar{j}}^A P_{i\bar{j}}^B
\end{aligned}$$

Which is used in the Orca implementation.

The use of the Local spin-implementation is very easy. All that is required is to divide the molecule into *fragments*. The rest happens automatically. For example, let us consider two nitrogen atoms at the dissociation limit. While the total spin state is S=0, the tow nitrogen atoms local exist in high-spin states (S=3/2). Consider the following test job:

```
! HF def2-SVP UHF TightSCF PModel

%scf brokensym 3,3 end

* xyz 0 1
N(1) 0 0 0
N(2) 0 0 1094
*
```

and the output:

```
-------------------
LOCAL SPIN ANALYSIS (Loewdin* projector)
-------------------


(1) A.E. Clark; E.R. Davison J. Chem. Phys. (2001), 115(16), pp 7382-7392
(2)  C. Herrmann, M. Reiher, B.A. Hess J. Chem. Phys. (2005) 122, art 034102-1

Number of fragments       = 2
Number of basis functions = 28
Number of atoms           = 2

   ... Fragment AO indices were mapped
   ... intra-fragment orthogonalization completed
   ... Global Loewdin orthogonalizer constructed
   ... Loewdin orthogonalized occupied orbitals constructed


 <SA*SB>        1            2
-----------------------------------
   1    :     3.7568
   2    :    -2.2500       3.7568


                <SzA>      Seff(A)
         --------------------------
   1    :     1.5000      1.5017
   2    :    -1.5000      1.5017
```

thus perfectly corresponding to the expectations. The same can be done at the CASSCF level:

```
! HF def2-SVP UHF TightSCF PModel

%casscf nel 6 norb 6 nroots 1 end

* xyz 0 1
N(1) 0 0 0
N(2) 0 0 1094
*
```

With the result:

```
<SA*SB>        1            2
-----------------------------------
   1    :     3.7500
   2    :    -3.7500       3.7500


               <SzA>*      Seff(A)
        --------------------------
   1    :       n.a.      1.5000
   2    :       n.a.      1.5000

* = for a singlet state all <SzA> values are zero by definition
```

Thus, cleanly confirming the expectations. In addition, if nroots > 1, the printing will contain the state-specific analysis of all roots.

As a less trivial example, consider a typical Fe(III) antiferromatically coupled transition metal dimer. An appropriate input may be:

```
! PBE def2-SV(P) TightSCF KDIIS SOSCF PModel

%scf
  brokensym 5,5
end


* xyz -2 1
Fe(1)        -1.93818         0.53739        -0.00010
Fe(2)         1.06735         0.47031         0.00029
S(3)        -0.38935         2.59862        -0.00983
S(3)        -0.48170        -1.59050         0.01091
S(1)         2.68798         0.43924         1.99710
S(1)         2.68692         0.42704        -1.99712
S(2)        -3.55594         0.56407        -1.99889
S(2)        -3.55850         0.58107         1.99646
H(1)         3.91984         0.39462         1.47608
H(1)         3.91940         0.39536        -1.47662
H(2)        -4.78410         0.69179        -1.48280
H(2)        -4.78991         0.49249         1.47983
*
```

Where one of the bridging sulfurs was assigned to each site respectively.

```
 <SA*SB>        1            2            3
-----------------------------------------------
   1    :     7.3346
   2    :    -4.5300       7.3349
   3    :    -0.7229      -0.7230       1.9403


               <SzA>      Seff(A)
```

(continues on next page)

(continued from previous page)

```
        -------------------------
   1    :     1.7561      2.2540
   2    :    -1.7562      2.2541
   3    :     0.0001      0.9800
```

The output from this calculation shows the expected results, with the local site spins being close to their ideal value 2.5 which would hold for a high-spin Fe(III) ion.

5.1.11 Population Analysis of UNOs

Population analyses for *UHF natural Orbitals* can be requested via the respective print options in the `%output` block:

```
%output
  Print[ P_UNO_OccNum      ] = 1;         # Occupation numbers
  Print[ P_UNO_AtPopMO_M   ] = 0;         # Mulliken atom pop. per UNO
  Print[ P_UNO_OrbPopMO_M] = 0;           # Mulliken orbital pop. per UNO
  Print[ P_UNO_ReducedOrbPopMO_M] = 0;    # Mulliken reduced orbital pop. per UNO
  Print[ P_UNO_AtPopMO_L   ] = 0;         # Loewdin atom pop. per UNO
  Print[ P_UNO_OrbPopMO_L] = 0;           # Loewdin orbital pop. per UNO
  Print[ P_UNO_ReducedOrbPopMO_L] = 0;    # Loewdin reduced orbital pop. per UNO
end
```

5.1.12 Keywords

A collection of useful keywords in the context of population analyses is given in Table 5.1, Table 5.2, Table 5.3, and Table 5.4.

Table 5.1: Simple input keywords for population analyses.

Keyword	Description
`AllPop`	Activates all population analyses
`NoPop`	Deactivates all population analyses
`ReducedPop`	Prints reduced orbital populations per MO
`NoReducedPop`	Deactivates printing of reduced orbital populations
`Mulliken`	Activates *Mulliken* population analysis
`NoMulliken`	Deactivates Mulliken population analysis
`Loewdin`	Activates *Löwdin* population analysis
`NoLoewdin`	Deactivates Löwdin population analysis
`Mayer`	Activates *Mayer* population analysis
`NoMayer`	Deactivates Mayer population analysis
`NPA`	Activates *NPA*. NBO program required
`NBO`	Activates *NPA* and *NBO* analyses. NBO program required
`NoNBO`	Deactivates NBO
`NoNPA`	Deactivates NPA and NBO
`MBIS`	Activates *MBIS* population analysis
`CHELPG`	Requests calculation of *CHELPG* charges
`CHELPG(LARGE)`	Requests calculation of *CHELPG* charges with large grid

Table 5.2: `%output` block input keywords and options for population analyses.

Keyword	Option	Description
`Print[ <option> ]`	`P_Mayer`	Print Mayer population analysis (Default = on)
	`P_NatPop`	Print Natural population analysis (Default = off)

continues on next page

Table 5.2 – continued from previous page

Keyword	Option	Description
	P_NPA	Print Natural population analysis (Default = off)
	P_Hirshfeld	Print Hirshfeld population analysis (Default = off)
	P_MBIS	Print MBIS population analysis (Default = off)
	P_Mulliken	Print Mulliken population analysis (Default = on)
	P_AtCharges_M	Print Mulliken atomic charges
	P_OrbCharges_M	Print Mulliken orbital charges
	P_FragCharges_M	Print Mulliken fragment charges
	P_FragBondOrder_M	Print Mulliken fragment bond orders
	P_BondOrder_M	Print Mulliken bond orders
	P_ReducedOrbPop_M	Print Mulliken reduced orbital charges
	P_FragPopMO_M	Print Mulliken fragment population for each MO
	P_FragOvlMO_M	Print Mulliken overlap populations per fragment pair
	P_AtPopMO_M	Print Mulliken atomic charges in each MO
	P_OrbPopMO_M	Print Mulliken orbital population for each MO
	P_ReducedOrbPopMO_M	Print Mulliken reduced orbital population for each MO
	P_Loewdin	Print Loewdin population analysis (Default = on)
	P_AtCharges_L	Print Loewdin atomic charges
	P_OrbCharges_L	Print Loewdin orbital charges
	P_FragCharges_L	Print Loewdin fragment charges
	P_FragBondOrder_L	Print Loewdin fragment bond orders
	P_BondOrder_L	Print Loewdin bond orders
	P_ReducedOrbPop_L	Print Loewdin reduced orbital charges
	P_FragPopMO_L	Print Loewdin fragment population for each MO
	P_FragOvlMO_L	Print Loewdin overlap populations per fragment pair
	P_AtPopMO_L	Print Loewdin atomic charges in each MO
	P_OrbPopMO_L	Print Loewdin orbital population for each MO
	P_ReducedOrbPopMO_L	Print Loewdin reduced orbital population for each MO
	P_Fragments	Print fragment information
	P_GUESSPOP	Print initial guess populations
	P_UNO_FragPopMO_M	Print Mulliken fragment population per UNO
	P_UNO_OrbPopMO_M	Print Mulliken orbital pop. per UNO
	P_UNO_AtPopMO_M	Print Mulliken atomic charges per UNO
	P_UNO_ReducedOrbPopMO_M	Print Mulliken reduced orbital pop. per UNO
	P_UNO_FragPopMO_L	Print Loewdin fragment population per UNO
	P_UNO_OrbPopMO_L	Print Loewdin orbital pop. per UNO
	P_UNO_AtPopMO_L	Print Loewdin atomic charges per UNO
	P_UNO_ReducedOrbPopMO_L	Print Loewdin reduced orbital pop. per UNO
	P_UNO_OccNum	Print occupation numbers per UNO

Table 5.3: `%method` block input keywords and options for population analyses.

Keyword	Option	Description
LOEWDIN_BONDORDE	<real>	Sets printing threshold for Löwdin bond orders (default = 0.05)
MAYER_BONDORDERT	<real>	Sets printing threshold for Mayer bond orders (default = 0.1)
MBIS_CHARGETHRES	<real>	Sets convergence threshold for MBIS charges (default = 10^{-6})
MBIS_LARGEPRINT	true/false	Activate printing of *MBIS Quantities* (default = false)
MBIS_ORIGIN_MULT	CenterOfCoor	Sets origin for MBIS properties to center of coordinate system (0,0,0)
	CenterOfMass	Sets origin for MBIS properties to center of mass
	CenterOfNucCl	Sets origin for MBIS properties to center of nuclear charge
	CenterXYZ	Sets origin for MBIS properties to arbitrary position, set coordinates with MBIS_ORIMULT_XYZ
	CenterOfEach	Sets origin for MBIS properties to center of each atom (default)
MBIS_ORIMULT_XYZ	<x,y,z>	Set the coordinates, otherwise 0,0,0 (unit: Angstrom)

Table 5.4: %chelpg block input keywords and options for CHELPG charges.

Keyword	Option	Description
GRID	<real>	Spacing of the regular grid in Å (default = 0.3)
RMAX	<real>	Maximum distance of all atoms to any gridpoint in Å (default = 2.8)
VDWRADII	COSMO	Activates usage of COSMO VdW radii of Atoms (default)
	BW	Activates usage of Breneman, Wiberg radii
DIPOLE	false/true	If true, the charges also reproduce the total dipole moment (default = false)

5.2 Natural Bond Orbital (NBO) Analysis

A popular and useful method for population analysis is the natural bond orbital analysis due to Weinhold and co-workers. It is implemented in the **NBO** program which is distributed in older versions via the CCL list and in newer versions via the University of Wisconsin/Madison. Information about the NBO program can be found at http://www.chem.wisc.edu/~nbo7. In order to use it together with ORCA you need a version of the stand-alone executable. Starting with version 3.1.x ORCA can only be used with NBO6 or NBO7. To specify the NBO executable the environment variable NBOEXE=/full/name/of/nbo7-executable has to be set. As the NBO part of the interface is not independent of the integer data-type width (i4 or i8), the NBO executable which will be used together with ORCA has to be compiled using i4!

ORCA features two methods to interface with the **NBO** program: ! NBO keyword and the %nbo-block. The following example illustrates the use for formaldehyde:

```
#
# Test the interface to the NBO program
#
! RHF SVP NBO

* xyz 0 1
  C     0.000000    0.000000    0.000000
  O     1.200000    0.000000    0.000000
  H    -0.550000    0.952628    0.000000
  H    -0.550000   -0.952628   -0.000000
*
```

This produces the following output:

```
Now starting NBO....

 ************************************ NBO 7.0 ************************************
            N A T U R A L   A T O M I C   O R B I T A L   A N D
         N A T U R A L   B O N D   O R B I T A L   A N A L Y S I S
 ************************* development version (D000000) *************************
  (c) Copyright 1996-2018 Board of Regents of the University of Wisconsin System
      on behalf of the Theoretical Chemistry Institute.  All rights reserved.

         Cite this program [NBO 7.0.0 (15-Nov-2018)] as:

         NBO 7.0.  E. D. Glendening, J. K. Badenhoop, A. E. Reed,
         J. E. Carpenter, J. A. Bohmann, C. M. Morales, P. Karafiloglou,
         C. R. Landis, and F. Weinhold, Theoretical Chemistry Institute,
         University of Wisconsin, Madison, WI (2018)

      /NPA    / : Natural Population Analysis
      /NBO    / : Natural Bond Orbital Analysis
      /AONBO  / : Checkpoint the AO to NBO transformation
      /ARCHIVE/ : Write the archive file to lfn47

 Job title:  ORCA Job: NBO_1


 NATURAL POPULATIONS:  Natural atomic orbital occupancies

  NAO Atom No lang   Type(AO)    Occupancy      Energy
 -----------------------------------------------------------
    1    C  1  s      Cor( 1s)     1.99997     -11.34329
    2    C  1  s      Val( 2s)     1.01533      -0.17540
    3    C  1  s      Ryd( 3s)     0.00701       0.61376
    4    C  1  px     Val( 2p)     0.81697       0.08822
    5    C  1  px     Ryd( 3p)     0.01268       0.63900
    6    C  1  py     Val( 2p)     1.09795      -0.01243
    7    C  1  py     Ryd( 3p)     0.00055       0.80803
    8    C  1  pz     Val( 2p)     0.66003      -0.03464
    9    C  1  pz     Ryd( 3p)     0.00283       0.62824
   10    C  1  dxy    Ryd( 3d)     0.00576       2.75039
   11    C  1  dxz    Ryd( 3d)     0.00375       2.25746
   12    C  1  dyz    Ryd( 3d)     0.00000       2.08566
   13    C  1  dx2y2  Ryd( 3d)     0.00337       2.74845
   14    C  1  dz2    Ryd( 3d)     0.00114       2.40647

   15    O  2  s      Cor( 1s)     1.99998     -20.56485
   16    O  2  s      Val( 2s)     1.70725      -0.92198
   17    O  2  s      Ryd( 3s)     0.00171       1.55322
   18    O  2  px     Val( 2p)     1.62177      -0.42255
   19    O  2  px     Ryd( 3p)     0.00079       1.29654
   20    O  2  py     Val( 2p)     1.91529      -0.46844
   21    O  2  py     Ryd( 3p)     0.00383       1.41052
   22    O  2  pz     Val( 2p)     1.32984      -0.28626
   23    O  2  pz     Ryd( 3p)     0.00011       1.30080
   24    O  2  dxy    Ryd( 3d)     0.00213       3.26414
   25    O  2  dxz    Ryd( 3d)     0.00340       3.20490
   26    O  2  dyz    Ryd( 3d)     0.00000       2.98918
   27    O  2  dx2y2  Ryd( 3d)     0.00406       3.55008
   28    O  2  dz2    Ryd( 3d)     0.00119       3.17511

   29    H  3  s      Val( 1s)     0.88576       0.07107
   30    H  3  s      Ryd( 2s)     0.00298       0.41181
   31    H  3  px     Ryd( 2p)     0.00030       2.18260
   32    H  3  py     Ryd( 2p)     0.00159       2.49146
```

(continues on next page)

(continued from previous page)

```
 33    H  3  pz     Ryd( 2p)     0.00002       1.85643

 34    H  4  s      Val( 1s)     0.88576       0.07107
 35    H  4  s      Ryd( 2s)     0.00298       0.41181
 36    H  4  px     Ryd( 2p)     0.00030       2.18260
 37    H  4  py     Ryd( 2p)     0.00159       2.49146
 38    H  4  pz     Ryd( 2p)     0.00002       1.85643


Summary of Natural Population Analysis:

                                     Natural Population
            Natural    -----------------------------------------------
 Atom No    Charge        Core      Valence    Rydberg      Total
-------------------------------------------------------------------
   C  1    0.37265      1.99997     3.59028    0.03709     5.62735
   O  2   -0.59134      1.99998     6.57415    0.01720     8.59134
   H  3    0.10934      0.00000     0.88576    0.00489     0.89066
   H  4    0.10934      0.00000     0.88576    0.00489     0.89066
===================================================================
* Total * -0.00000      3.99995    11.93596    0.06408    16.00000

                                Natural Population
-------------------------------------------------------
  Core                      3.99995 ( 99.9988% of    4)
  Valence                  11.93596 ( 99.4664% of   12)
  Natural Minimal Basis    15.93592 ( 99.5995% of   16)
  Natural Rydberg Basis     0.06408 (  0.4005% of   16)
-------------------------------------------------------

   Atom No         Natural Electron Configuration
-----------------------------------------------------------------------------
     C  1      [core]2s( 1.02)2p( 2.57)3s( 0.01)3p( 0.02)3d( 0.01)
     O  2      [core]2s( 1.71)2p( 4.87)3d( 0.01)
     H  3            1s( 0.89)
     H  4            1s( 0.89)


NATURAL BOND ORBITAL ANALYSIS:

                           Occupancies       Lewis Structure    Low   High
        Max    Occ     -------------------  -----------------   occ   occ
 Cycle  Ctr   Thresh    Lewis   non-Lewis     CR  BD  nC  LP    (L)   (NL)
=============================================================================
   1     2     1.90    15.89671   0.10329      2   4   0   2     0      0
-----------------------------------------------------------------------------

Structure accepted: No low occupancy Lewis orbitals

-------------------------------------------------------
  Core                      3.99995 ( 99.999% of   4)
  Valence Lewis            11.89676 ( 99.140% of  12)
 ==================      ============================
  Total Lewis              15.89671 ( 99.354% of  16)
 -------------------------------------------------------
  Valence non-Lewis         0.07835 (  0.490% of  16)
  Rydberg non-Lewis         0.02493 (  0.156% of  16)
 ==================      ============================
  Total non-Lewis           0.10329 (  0.646% of  16)
-------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
    (Occupancy)   Bond orbital / Coefficients / Hybrids
------------------ Lewis --------------------------------------------------------
  1. (1.99997) CR ( 1) C  1            s(100.00%)
                                        1.0000  0.0000  0.0000  0.0000  0.0000
                                        0.0000  0.0000  0.0000  0.0000  0.0000
                                        0.0000  0.0000  0.0000  0.0000
  2. (1.99998) CR ( 1) O  2            s(100.00%)
                                        1.0000  0.0000  0.0000  0.0000  0.0000
                                        0.0000  0.0000  0.0000  0.0000  0.0000
                                        0.0000  0.0000  0.0000  0.0000
  3. (1.98853) LP ( 1) O  2            s( 56.22%)p 0.78( 43.73%)d 0.00(  0.05%)
                                        0.0000  0.7496 -0.0170  0.6612  0.0069
                                        0.0000  0.0000  0.0000  0.0000  0.0000
                                        0.0000  0.0000 -0.0201  0.0100
  4. (1.91757) LP ( 2) O  2            s(  0.00%)p 1.00( 99.89%)d 0.00(  0.11%)
                                        0.0000  0.0000  0.0000 -0.0000 -0.0000
                                        0.9994 -0.0098  0.0000 -0.0000 -0.0330
                                       -0.0000  0.0000  0.0000 -0.0000
  5. (1.99996) BD ( 1) C  1- O  2
             ( 33.33%)   0.5773* C  1 s(  0.00%)p 1.00( 99.44%)d 0.01(  0.56%)
                                        0.0000 -0.0000 -0.0000  0.0000 -0.0000
                                       -0.0000  0.0000  0.9951 -0.0652 -0.0000
                                        0.0750  0.0000 -0.0000  0.0000
             ( 66.67%)   0.8165* O  2 s(  0.00%)p 1.00( 99.75%)d 0.00(  0.25%)
                                        0.0000 -0.0000  0.0000  0.0000 -0.0000
                                       -0.0000  0.0000  0.9987 -0.0090  0.0000
                                       -0.0505  0.0000  0.0000 -0.0000
  6. (1.99975) BD ( 2) C  1- O  2
             ( 32.59%)   0.5709* C  1 s( 32.18%)p 2.09( 67.35%)d 0.01(  0.46%)
                                        0.0000  0.5628  0.0714  0.8149  0.0973
                                       -0.0000 -0.0000 -0.0000 -0.0000 -0.0000
                                       -0.0000  0.0000  0.0618 -0.0286
             ( 67.41%)   0.8211* O  2 s( 43.84%)p 1.27( 55.85%)d 0.01(  0.31%)
                                        0.0000  0.6615  0.0284 -0.7470 -0.0215
                                       -0.0000  0.0000 -0.0000 -0.0000  0.0000
                                        0.0000  0.0000  0.0490 -0.0270
  7. (1.99548) BD ( 1) C  1- H  3
             ( 56.63%)   0.7526* C  1 s( 33.98%)p 1.94( 65.86%)d 0.00(  0.16%)
                                        0.0000  0.5826 -0.0192 -0.3995 -0.0029
                                        0.7063 -0.0087  0.0000 -0.0000 -0.0318
                                       -0.0000  0.0000 -0.0180 -0.0153
             ( 43.37%)   0.6585* H  3 s( 99.79%)p 0.00(  0.21%)
                                        0.9989 -0.0095  0.0184 -0.0416  0.0000
  8. (1.99548) BD ( 1) C  1- H  4
             ( 56.63%)   0.7526* C  1 s( 33.98%)p 1.94( 65.86%)d 0.00(  0.16%)
                                        0.0000  0.5826 -0.0192 -0.3995 -0.0029
                                       -0.7063  0.0087 -0.0000  0.0000  0.0318
                                        0.0000  0.0000 -0.0180 -0.0153
             ( 43.37%)   0.6585* H  4 s( 99.79%)p 0.00(  0.21%)
                                        0.9989 -0.0095  0.0184  0.0416 -0.0000
---------------- non-Lewis ----------------------------------------------------
  9. (0.00000) BD*( 1) C  1- O  2
             ( 66.67%)   0.8165* C  1 s(  0.00%)p 1.00( 99.44%)d 0.01(  0.56%)
             ( 33.33%)  -0.5773* O  2 s(  0.00%)p 1.00( 99.75%)d 0.00(  0.25%)
 10. (0.00000) BD*( 2) C  1- O  2
             ( 67.41%)   0.8211* C  1 s( 32.18%)p 2.09( 67.35%)d 0.01(  0.46%)
             ( 32.59%)  -0.5709* O  2 s( 43.84%)p 1.27( 55.85%)d 0.01(  0.31%)
 11. (0.03918) BD*( 1) C  1- H  3
             ( 43.37%)   0.6585* C  1 s( 33.98%)p 1.94( 65.86%)d 0.00(  0.16%)
                                        0.0000 -0.5826  0.0192  0.3995  0.0029
```

(continues on next page)

(continued from previous page)

```
                                        -0.7063  0.0087 -0.0000  0.0000  0.0318
                                         0.0000  0.0000  0.0180  0.0153
              ( 56.63%)  -0.7526* H  3 s( 99.79%)p 0.00(  0.21%)
                                        -0.9989  0.0095 -0.0184  0.0416 -0.0000
 12. (0.03918) BD*( 1) C  1- H  4
              ( 43.37%)   0.6585* C  1 s( 33.98%)p 1.94( 65.86%)d 0.00(  0.16%)
                                         0.0000 -0.5826  0.0192  0.3995  0.0029
                                         0.7063 -0.0087  0.0000 -0.0000 -0.0318
                                        -0.0000  0.0000  0.0180  0.0153
              ( 56.63%)  -0.7526* H  4 s( 99.79%)p 0.00(  0.21%)
                                        -0.9989  0.0095 -0.0184 -0.0416  0.0000
 13. (0.00969) RY ( 1) C  1            s( 29.83%)p 2.30( 68.57%)d 0.05(  1.60%)
                                         0.0000 -0.0565  0.5432 -0.1169  0.8198
                                         0.0000 -0.0000  0.0000 -0.0000 -0.0000
                                         0.0000  0.0000  0.1087 -0.0648
 14. (0.00517) RY ( 2) C  1            s(  0.00%)p 1.00(  9.56%)d 9.46( 90.44%)
                                         0.0000  0.0000 -0.0000  0.0000 -0.0000
                                         0.0465  0.3057  0.0000 -0.0000  0.9510
                                         0.0000  0.0000 -0.0000 -0.0000
 15. (0.00001) RY ( 3) C  1            s( 20.02%)p 0.82( 16.47%)d 3.17( 63.52%)
 16. (0.00000) RY ( 4) C  1            s(  0.00%)p 1.00( 90.64%)d 0.10(  9.36%)
 17. (0.00000) RY ( 5) C  1            s(  0.00%)p 1.00(100.00%)d 0.00(  0.00%)
 18. (0.00000) RY ( 6) C  1            s( 42.72%)p 0.35( 15.02%)d 0.99( 42.26%)
 19. (0.00000) RY ( 7) C  1            s(  0.00%)p 1.00(  0.56%)d99.99( 99.44%)
 20. (0.00000) RY ( 8) C  1            s(  0.00%)p 0.00(  0.00%)d 1.00(100.00%)
 21. (0.00000) RY ( 9) C  1            s(  7.29%)p 0.09(  0.66%)d12.63( 92.05%)
 22. (0.00368) RY ( 1) O  2            s(  0.00%)p 1.00( 98.96%)d 0.01(  1.04%)
                                         0.0000 -0.0000 -0.0000  0.0000  0.0000
                                         0.0064  0.9948 -0.0000  0.0000 -0.1018
                                         0.0000  0.0000 -0.0000  0.0000
 23. (0.00014) RY ( 2) O  2            s( 35.10%)p 1.44( 50.60%)d 0.41( 14.30%)
                                         0.0000 -0.0178  0.5922  0.0556 -0.7091
                                         0.0000 -0.0000 -0.0000  0.0000 -0.0000
                                         0.0000  0.0000  0.3336 -0.1780
 24. (0.00000) RY ( 3) O  2            s( 56.05%)p 0.25( 13.79%)d 0.54( 30.17%)
 25. (0.00000) RY ( 4) O  2            s(  0.00%)p 1.00(100.00%)d 0.00(  0.00%)
 26. (0.00000) RY ( 5) O  2            s(  0.00%)p 1.00(  1.14%)d86.35( 98.86%)
 27. (0.00000) RY ( 6) O  2            s(  0.00%)p 1.00(  0.25%)d99.99( 99.75%)
 28. (0.00000) RY ( 7) O  2            s(  0.00%)p 0.00(  0.00%)d 1.00(100.00%)
 29. (0.00000) RY ( 8) O  2            s(  6.72%)p 5.27( 35.42%)d 8.61( 57.85%)
 30. (0.00000) RY ( 9) O  2            s(  2.07%)p 0.30(  0.61%)d47.00( 97.32%)
 31. (0.00308) RY ( 1) H  3            s( 99.42%)p 0.01(  0.58%)
                                         0.0096  0.9970 -0.0710 -0.0281  0.0000
 32. (0.00002) RY ( 2) H  3            s(  0.22%)p99.99( 99.78%)
 33. (0.00002) RY ( 3) H  3            s(  0.00%)p 1.00(100.00%)
 34. (0.00001) RY ( 4) H  3            s(  0.57%)p99.99( 99.43%)
 35. (0.00308) RY ( 1) H  4            s( 99.42%)p 0.01(  0.58%)
                                         0.0096  0.9970 -0.0710  0.0281  0.0000
 36. (0.00002) RY ( 2) H  4            s(  0.22%)p99.99( 99.78%)
 37. (0.00002) RY ( 3) H  4            s(  0.00%)p 1.00(100.00%)
 38. (0.00001) RY ( 4) H  4            s(  0.57%)p99.99( 99.43%)


NHO DIRECTIONALITY AND BOND BENDING (deviation from line of nuclear centers at
                                     the position of maximum hybrid amplitude)

        [Thresholds for printing:  angular deviation  >  1.0 degree]
                                   p- or d-character  > 25.0%
                                   orbital occupancy  >  0.10e

                       Line of Centers        Hybrid 1             Hybrid 2
```

(continues on next page)

(continued from previous page)

```
                        ---------------  -------------------  ------------------
          NBO            Theta   Phi    Theta   Phi    Dev   Theta   Phi    Dev
================================================================================
  3. LP ( 1) O  2          --     --     90.0    0.0   --      --     --    --
  4. LP ( 2) O  2          --     --     90.0   90.7   --      --     --    --
  5. BD ( 1) C  1- O  2   90.0    0.0     3.0    0.0  87.0   178.7  180.0  88.7


SECOND ORDER PERTURBATION THEORY ANALYSIS OF FOCK MATRIX IN NBO BASIS

    Threshold for printing:   0.50 kcal/mol
                                                                E(2) E(NL)-E(L) F(L,NL)
     Donor (L) NBO              Acceptor (NL) NBO      kcal/mol   a.u.    a.u.
================================================================================

within unit  1
   3. LP ( 1) O  2            13. RY ( 1) C  1            8.58    1.40   0.098
   4. LP ( 2) O  2            11. BD*( 1) C  1- H  3     26.11    1.16   0.156
   4. LP ( 2) O  2            12. BD*( 1) C  1- H  4     26.11    1.16   0.156
   4. LP ( 2) O  2            14. RY ( 2) C  1            5.72    3.06   0.118
   4. LP ( 2) O  2            26. RY ( 5) O  2            0.73    3.75   0.047
   7. BD ( 1) C  1- H  3      12. BD*( 1) C  1- H  4      0.74    1.42   0.029
   7. BD ( 1) C  1- H  3      22. RY ( 1) O  2            2.25    2.12   0.062
   8. BD ( 1) C  1- H  4      11. BD*( 1) C  1- H  3      0.74    1.42   0.029
   8. BD ( 1) C  1- H  4      22. RY ( 1) O  2            2.25    2.12   0.062


NATURAL BOND ORBITALS (Summary):

                                                      Principal Delocalizations
          NBO                 Occupancy    Energy   (geminal,vicinal,remote)
================================================================================
Molecular unit  1  (CH2O)
------ Lewis -------------------------------------
   1. CR ( 1) C  1             1.99997   -11.34329
   2. CR ( 1) O  2             1.99998   -20.56485
   3. LP ( 1) O  2             1.98853    -0.81352  13(v)
   4. LP ( 2) O  2             1.91757    -0.46975  11(v),12(v),14(v),26(g)
   5. BD ( 1) C  1- O  2       1.99996    -0.53505
   6. BD ( 2) C  1- O  2       1.99975    -1.23345
   7. BD ( 1) C  1- H  3       1.99548    -0.72703  22(v),12(g)
   8. BD ( 1) C  1- H  4       1.99548    -0.72703  22(v),11(g)
------ non-Lewis ----------------------------------
   9. BD*( 1) C  1- O  2       0.00000     0.20704
  10. BD*( 2) C  1- O  2       0.00000     0.95146
  11. BD*( 1) C  1- H  3       0.03918     0.69317
  12. BD*( 1) C  1- H  4       0.03918     0.69317
  13. RY ( 1) C  1             0.00969     0.58169
  14. RY ( 2) C  1             0.00517     2.58837
  15. RY ( 3) C  1             0.00001     1.75652
  16. RY ( 4) C  1             0.00000     0.96046
  17. RY ( 5) C  1             0.00000     0.64510
  18. RY ( 6) C  1             0.00000     1.49223
  19. RY ( 7) C  1             0.00000     2.24615
  20. RY ( 8) C  1             0.00000     2.08566
  21. RY ( 9) C  1             0.00000     2.49414
  22. RY ( 1) O  2             0.00368     1.39676
  23. RY ( 2) O  2             0.00014     1.56107
  24. RY ( 3) O  2             0.00000     2.18160
  25. RY ( 4) O  2             0.00000     1.30222
  26. RY ( 5) O  2             0.00000     3.27920
```

(continues on next page)

(continued from previous page)

```
   27. RY ( 6) O  2             0.00000     3.20505
   28. RY ( 7) O  2             0.00000     2.98918
   29. RY ( 8) O  2             0.00000     2.69359
   30. RY ( 9) O  2             0.00000     3.12898
   31. RY ( 1) H  3             0.00308     0.41874
   32. RY ( 2) H  3             0.00002     2.57996
   33. RY ( 3) H  3             0.00002     1.85643
   34. RY ( 4) H  3             0.00001     2.06898
   35. RY ( 1) H  4             0.00308     0.41874
   36. RY ( 2) H  4             0.00002     2.57996
   37. RY ( 3) H  4             0.00002     1.85643
   38. RY ( 4) H  4             0.00001     2.06898
          -------------------------------
                 Total Lewis   15.89671  ( 99.3545%)
           Valence non-Lewis    0.07835  (  0.4897%)
           Rydberg non-Lewis    0.02493  (  0.1558%)
          -------------------------------
               Total unit  1   16.00000  (100.0000%)
              Charge unit  1    0.00000

 $CHOOSE
   LONE 2 2 END
   BOND D 1 2 S 1 3 S 1 4 END
 $END

 NBO analysis completed in 0.05 CPU seconds (0 wall seconds)
 Maximum scratch memory used by NBO was 297106 words (2.27 MB)
Stopping NBO...Storing NBOs: NBO_1.nbo

                                 *** returned from  NBO  program ***
```

Thus, in this example the NBO analysis of formaldehyde shows that a single Lewis structure is dominant with single bonds between C and H, a double bond between C and O and two lone pairs at the oxygen – just as ordinary chemical arguments would imply. In addition, the program produces the four corresponding valence antibonds. The remaining components of the basis set span the "Rydberg" space and lead to semilocalized, orthogonal orbitals that are assigned to single atoms (Note the nomenclature: BD = bond, BD* = antibond, LP = lone pair, CR = core orbital, RY= Rydberg orbital). The NPA analysis shows a patially negative oxygen and partially positive carbon and hydrogen atoms.

Additionally, the NBO orbitals are stored in the ORCA .gbw file format as `jobname.nbo`. This file can be used for further analysis and usage with ORCA e.g. for plotting orbitals via `orca_plot`.

The **NBO** program has many additional features and analysis tools. The features that are implemented in ORCA can be controlled via the `%nbo`-block

```
%nbo
NBOKEYLIST    = "$NBO     ... $END"
DELKEYLIST    = "$DEL     ... $END"
COREKEYLIST   = "$CORE    ... $END"
NRTSTRKEYLIST = "$NRTSTR  ... $END"
NPEPAKEYLIST  = "$NPEPA   ... $END"
end
```

The syntax of the respective keylists is given by the NBO6.x/NBO7.x manual.

Specifying the single `! NBO` keyword corresponds to the `%nbo`-block

```
%nbo
NBOKEYLIST    = "$NBO NBO NPA AONBO=C ARCHIVE $END"
end
```

The full set of features beyond those which can be give via the `%nbo` block can be accessed using the file `FILE.47`,

which is generated by the NBO program. This is an ascii file that can be edited with a text editor. Add or remove keywords in the corresponding blocks as needed and call the gennbo program like

```
gennbo < FILE.47 >jobname.nboout
```

(a) (a) π (b) (b) π - LP (c) (c) π*

(d) (d) σ (e) (e) σ - LP (f) (f) σ*

Fig. 5.1: Six NBOs of the H_2CO molecule. Shown are the occupied bonding π and σ orbitals (left) for C and O, the two oxygen lone pairs (middle) and the two π^* and σ^* antibonding orbitals (right).

The `FILE.47` file looks like:

```
$GENNBO  NATOMS=4  NBAS=38  UPPER  BODM  FORMAT  $END
 $NBO  $END
 $COORD
 ORCA Job: check
     6    6       0.000000       0.000000       0.000000
     8    8       2.267671       0.000000       0.000000
     1    1      -1.039349       1.800206       0.000000
     1    1      -1.039349      -1.800206       0.000000
 $END
 $BASIS
```

If you have no need for this (rather large) file, then you have to delete it manually!

5.2.1 NBO Deletions

An advanced feature, which has been implemented via the ORCA-NBO interface, is the possibility of using deletions.

```
! RHF 3-21G BOHRS TightSCF

%nbo
nbokeylist="$nbo nbo npa aonbo=c archive $end"
delkeylist="$del lewis delete 1 element 3 11 $end"
end


*xyz 0 1
 C              1.4089705283        0.0210567401        0.0000000000
 N             -1.3645072652       -0.1355759321        0.0000000000
 H              1.9849776453        1.9986808971        0.0000000000
 H              2.1492280974       -0.9096841007        1.6818209547
 H              2.1492280974       -0.9096841007       -1.6818209547
 H             -2.0504340036        0.7268536543       -1.5583845544
 H             -2.0504340036        0.7268536543        1.5583845544
*
```

The `DELKEYLIST` provides NBO with the task to perform certain deletions of orbitals/interactions. Per deletion ORCA calculates a new Fock matrix on basis of an NBO density corresponding to the deletions:

```
Stopping NBO...Starting NBO again for $del instructions...

 LEWIS: Delete all non-Lewis NBOs
 Deletion of the following orbitals from the NBO Fock matrix:
   10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28

 Orbital occupancies:

       Orbital                          No deletions  This deletion    Change
 ---------------------------------------------------------------------------
    1. CR ( 1) C  1                       1.99978        2.00000       0.00022
    2. CR ( 1) N  2                       1.99983        2.00000       0.00017
    3. LP ( 1) N  2                       1.97796        2.00000       0.02204
    4. BD ( 1) C  1- N  2                 1.99846        2.00000       0.00154
    5. BD ( 1) C  1- H  3                 1.99858        2.00000       0.00142
    6. BD ( 1) C  1- H  4                 1.99406        2.00000       0.00594
    7. BD ( 1) C  1- H  5                 1.99406        2.00000       0.00594
    8. BD ( 1) N  2- H  6                 1.99440        2.00000       0.00560
    9. BD ( 1) N  2- H  7                 1.99440        2.00000       0.00560
   10. BD*( 1) C  1- N  2                 0.00009        0.00000      -0.00009
   11. BD*( 1) C  1- H  3                 0.01567        0.00000      -0.01567
   12. BD*( 1) C  1- H  4                 0.00763        0.00000      -0.00763
   13. BD*( 1) C  1- H  5                 0.00763        0.00000      -0.00763
   14. BD*( 1) N  2- H  6                 0.00424        0.00000      -0.00424
   15. BD*( 1) N  2- H  7                 0.00424        0.00000      -0.00424
   16. RY ( 1) C  1                       0.00094        0.00000      -0.00094
   17. RY ( 2) C  1                       0.00034        0.00000      -0.00034
   18. RY ( 3) C  1                       0.00020        0.00000      -0.00020
   19. RY ( 4) C  1                       0.00001        0.00000      -0.00001
   20. RY ( 1) N  2                       0.00114        0.00000      -0.00114
   21. RY ( 2) N  2                       0.00044        0.00000      -0.00044
   22. RY ( 3) N  2                       0.00034        0.00000      -0.00034
   23. RY ( 4) N  2                       0.00001        0.00000      -0.00001
   24. RY ( 1) H  3                       0.00163        0.00000      -0.00163
   25. RY ( 1) H  4                       0.00079        0.00000      -0.00079
   26. RY ( 1) H  5                       0.00079        0.00000      -0.00079
   27. RY ( 1) H  6                       0.00117        0.00000      -0.00117
```

(continues on next page)

(continued from previous page)

```
   28. RY ( 1) H  7                      0.00117        0.00000      -0.00117

 NEXT STEP:  Perform one SCF cycle to evaluate the energy of the new density
             matrix constructed from the deleted NBO Fock matrix.

 ------------------------------------------------------------------------------
Copying NBO density...
Calculating new Fock-Matrix...
Calculating Fock-Matrix...done!

New NBO energy via Fock-Matrix: -94.629937


Starting NBO again for $del/return energy instructions...

 ------------------------------------------------------------------------------
   Energy of deletion :        -94.629936711
     Total SCF energy :        -94.679444929
                          -------------------
       Energy change :          0.049508 a.u.,          31.067 kcal/mol
 ------------------------------------------------------------------------------
```

Multiple deletions can also be specified, as can be seen for this example. The output then also contains the additional energy values:

```
Starting NBO again for $del instructions...

 Deletion of the following NBO Fock matrix elements:
     3, 11;

 Orbital occupancies:

       Orbital                         No deletions  This deletion    Change
 ------------------------------------------------------------------------------
    1. CR ( 1) C  1                      1.99978        1.99978      -0.00000
    2. CR ( 1) N  2                      1.99983        1.99983      -0.00000
    3. LP ( 1) N  2                      1.97796        1.99348       0.01552
    4. BD ( 1) C  1- N  2                1.99846        1.99860       0.00015
    5. BD ( 1) C  1- H  3                1.99858        1.99845      -0.00014
    6. BD ( 1) C  1- H  4                1.99406        1.99404      -0.00002
    7. BD ( 1) C  1- H  5                1.99406        1.99404      -0.00002
    8. BD ( 1) N  2- H  6                1.99440        1.99450       0.00011
    9. BD ( 1) N  2- H  7                1.99440        1.99450       0.00011
   10. BD*( 1) C  1- N  2                0.00009        0.00008      -0.00000
   11. BD*( 1) C  1- H  3                0.01567        0.00042      -0.01525
   12. BD*( 1) C  1- H  4                0.00763        0.00780       0.00017
   13. BD*( 1) C  1- H  5                0.00763        0.00780       0.00017
   14. BD*( 1) N  2- H  6                0.00424        0.00424       0.00000
   15. BD*( 1) N  2- H  7                0.00424        0.00424       0.00000
   16. RY ( 1) C  1                      0.00094        0.00063      -0.00031
   17. RY ( 2) C  1                      0.00034        0.00034       0.00000
   18. RY ( 3) C  1                      0.00020        0.00032       0.00012
   19. RY ( 4) C  1                      0.00001        0.00002       0.00001
   20. RY ( 1) N  2                      0.00114        0.00115       0.00001
   21. RY ( 2) N  2                      0.00044        0.00044       0.00000
   22. RY ( 3) N  2                      0.00034        0.00034       0.00000
   23. RY ( 4) N  2                      0.00001        0.00001       0.00000
   24. RY ( 1) H  3                      0.00163        0.00092      -0.00072
   25. RY ( 1) H  4                      0.00079        0.00083       0.00005
   26. RY ( 1) H  5                      0.00079        0.00083       0.00005
   27. RY ( 1) H  6                      0.00117        0.00118       0.00000
```

(continues on next page)

(continued from previous page)

```
   28. RY ( 1) H  7                      0.00117        0.00118        0.00000

 NEXT STEP:  Perform one SCF cycle to evaluate the energy of the new density
             matrix constructed from the deleted NBO Fock matrix.

 ------------------------------------------------------------------------------
Copying NBO density...
Calculating new Fock-Matrix...
Calculating Fock-Matrix...done!

New NBO energy via Fock-Matrix: -94.668383


Starting NBO again for $del/return energy instructions...

 ------------------------------------------------------------------------------
   Energy of deletion :        -94.668383268
     Total SCF energy :        -94.679444929
                          -------------------
       Energy change :          0.011062 a.u.,           6.941 kcal/mol
 ------------------------------------------------------------------------------
```

Note

Note that deletions are only implemented for SCF methods!

5.2.2 NBO for Post-HF Densities

NBO analysis can be performed on all methods producing a density. In some methods the density generation has to be specified explictly, e. g. for MP2 calculations this would be:

```
! MP2 3-21G TightSCF BOHRS NBO

%MP2
density relaxed
end

*xyz 0 1
 C             1.4089705283        0.0210567401        0.0000000000
 N            -1.3645072652       -0.1355759321        0.0000000000
 H             1.9849776453        1.9986808971        0.0000000000
 H             2.1492280974       -0.9096841007        1.6818209547
 H             2.1492280974       -0.9096841007       -1.6818209547
 H            -2.0504340036        0.7268536543       -1.5583845544
 H            -2.0504340036        0.7268536543        1.5583845544
*
```

The output will contain both the NBO analysis of the SCF density as well as of the MP2 relaxed density. An NBO analysis of a density generated by the MDCI module can be specified as follows:

```
! CISD 3-21G TightSCF BOHRS NBO

%mdci
    density linearized
end

*xyz 0 1
 C        1.4089705283        0.0210567401        0.0000000000
```

(continues on next page)

(continued from previous page)

```
 N       -1.3645072652       -0.1355759321        0.0000000000
 H        1.9849776453        1.9986808971        0.0000000000
 H        2.1492280974       -0.9096841007        1.6818209547
 H        2.1492280974       -0.9096841007       -1.6818209547
 H       -2.0504340036        0.7268536543       -1.5583845544
 H       -2.0504340036        0.7268536543        1.5583845544
*
```

Again, the output will contain both the NBO analysis of the SCF density as well as of the CISD linearized density.

5.2.3 Natural Chemical Shielding Analysis (NCS)

For closed-shell calculations of NMR chemical shielding at the SCF level (see sections *NMR Chemical Shifts* and *EPRNMR - keywords for magnetic properties*), the NCS analysis can be requested by adding `NCS` to the `NBOKEYLIST`. The `NCS` keyword accepts the arguments `U`, `I`, `CSA`, `XYZ`, and `MO` to analyze the "unperturbed", "induced", anisotropic, Cartesian, and canonical MO contributions to the shielding tensors, respectively, as well as a decimal number for the printing threshold (in ppm). For more information, consult the NBO manual and the original publication [587].

```
! PBE def2-TZVP NMR
%nbo
  NBOKeyList = "$NBO NCS=0.01,I,U,XYZ $END"
end
* xyz 0 1
  H     0.00     0.00     0.00
  C     1.06     0.00     0.00
  N     2.23     0.00     0.00
*
```

```
Summary of isotropic NMR chemical shielding
 Total Lewis (L) and non-Lewis (NL) contributions: (ppm)

        NBO            H  1    C  2    N  3
 ---------------     ------- ------- -------
   1.  C 2(cr)   L    -0.18  200.26    0.18
                 NL   -0.02    0.02    0.00
   2.  N 3(cr)   L    -0.03   -0.12  235.18
                 NL    0.00    0.01    0.02
   3.  N 3(lp)   L     1.02  -33.00 -151.92
                 NL   -1.04    1.81   12.18
   4.  H 1- C 2  L    25.75  -49.28  -20.30
                 NL   -1.24    6.10    2.26
   5.  C 2- N 3  L     2.29   15.40   13.66
                 NL    0.02    0.00   -0.00
   6.  C 2- N 3  L     2.29   15.40   13.66
                 NL    0.02    0.00   -0.00
   7.  C 2- N 3  L     0.46  -77.94 -151.00
                 NL    0.05   -4.41    0.95
 ---------------     ------- ------- -------
           Lewis      31.59   70.70  -60.53
       non-Lewis      -2.21    3.53   15.42
 ---------------     ------- ------- -------
           Total      29.38   74.23  -45.11
```

5.3 Fractional Occupation Number Weighted Density (FOD)

Many approximate QC methods do not yield reliable results for systems with significant static electron correlation (SEC) and moreover it is often difficult to predict if the system in question suffers from SEC or not. Existing scalar SEC diagnostics (e.g., the T_1 diagnostic) do not provide any information where the SEC is located in the molecule. Furthermore, often quite expensive calculations have to be performed first (e.g., CCSD) in order to judge the reliability of the results based on a single value. Molecular systems with strong SEC (e.g. covalent bond-breaking, biradicals, open-shell transition metal complexes) are usually characterized by small energy gaps between frontier orbitals, and hence, the existence of many equally important determinants in their electronic wavefunction. This finding is used in the fractional occupation number weighted density (FOD) analysis proposed by Hansen and Grimme.[588] The FOD analysis is based on *finite temperature KS-DFT* where the *fractional occupation numbers* are determined from the Fermi distribution ("Fermi smearing").

$$f_i = \frac{1}{e^{(\varepsilon_i - E_F)/kT_{el}} + 1} \tag{5.25}$$

The central quantity of the FOD analysis is the fractional occupation number weighted electron density (ρ^{FOD}), a real-space function of the position vector r:

$$\rho^{FOD}(r) = \sum_i^N (\delta_1 - \delta_2 f_i)|\varphi_i(r)|^2 \tag{5.26}$$

(δ_1 and δ_2 are unity if the level is lower than E_F while they are 0 and -1, respectively, for levels higher than E_F). The f_i represent the fractional occupation numbers ($0 \leq f_i \leq 1$; sum over all electronic single-particle levels obtained by solving self-consistently the KS-SCF equations minimizing the *free*-electronic energy).

$\rho^{FOD}(r)$ can be plotted using a pre-defined contour surface value (see Section *FOD Plots*). The integration of ρ^{FOD} over all space yields as additional information a single size-extensive number termed N_{FOD} which correlates well with other scalar SEC diagnostics and can be used to globally quantify SEC effects in the molecule. Accordingly, FOD analysis and plots represent a cost-effective and robust way to identify the 'hot' (strongly correlated) electrons in a molecule and to choose appropriate approximate quantum chemical methods for a subsequent computational study of the systems in question. Based on our experience, the following rules of thumb can be derived:

- no significant ρ^{FOD}: use (double)-hybrid functionals or (DLPNO-)CCSD(T) (single-reference electronic structure)
- significant but rather localized ρ^{FOD}: use semi-local GGA functionals (or hybrid functional with low Fock-exchange, avoid HF or MP2; slight multi-reference character)
- significant *and* delocalized ρ^{FOD}: use multi-reference methods (or finite temperature DFT; strong multi-reference character)

Warning

Even though the FOD is intuitive and easy-to-use, it is not a 100% safe diagnostic for MR character and should only be used as a first indication. We recommend to always combine the FOD analysis with more sophisticated multi-reference diagnostics.

Tip

Mulliken reduced orbital charges based on $\rho^{FOD}(r)$ (see *Mulliken Population Analysis*) offer a fast alternative to get the information of the FOD plot (cf. *Example: p-Benzyne*).

5.3.1 Basic Usage

The FOD analysis is a very efficient and practicable tool to get information about the amount and localization of SEC in the system of question and can be invoked via the `!FOD` simple input keyword.

```
! FOD
```

This will perform the analysis at the default TPSS/def2-TZVP (`TightSCF`) level that was chosen since it is fast and robust. Besides the standard analysis output (cf. Section 5.3.3), ρ^{FOD} is stored in the file `basename.scfp_fod` file which is included in the general `basename.densities` container (for plotting the FOD, see Section 5.3.2).

The FOD analysis can be customized via simple input and the `%scf` block.

```
! B3LYP def2-TZVP TightSCF

%scf
 SmearTemp 9000
end
```

Important

- ρ^{FOD} (and N_{FOD}) strongly depend on the orbital energy gap which itself depends almost linearly on the amount of the non-local Fock exchange admixture a_x (cf. *hybrid DFT*). The following (empirical) function of the optimal electronic temperature T_{el} on a_x

$$T_{el} = 20000\,\mathrm{K} \times a_x + 5000\,\mathrm{K} \tag{5.27}$$

  is used to ensure that similar results of the FOD analysis are obtained with various functionals. For example, the `SmearTemp` has to be 5000 K for TPSS (a_x = 0), 9000 K for B3LYP (a_x = 20%), 10000 K for PBE0 (a_x = 25%), and 15800 K for M06-2x (a_x = 54%).

Some Notes on FOD

- The FOD analysis is not strongly dependent on the employed basis set (see supplementary information of the original publication[588]).
- The FOD analysis will be always printed (including Mulliken reduced orbital charges based on ρ^{FOD}) if `SmearTemp` $>$ 0 K.
- Since the $\hat{S}^2$ expectation value is not defined for fractional occupation numbers, its printout is omitted.
- The FOD analysis may also be useful for finding a suitable active space for e.g. in *CASSCF* calculations.

5.3.2 FOD Plots

The fractional occupation number weighted electron density (ρ^{FOD}, see *Fractional Occupation Number Weighted Density (FOD)*) can be plotted in 3D for a pre-defined contour surface value which, after extensive testing, was set to the default value of $\sigma = 0.005$ e/Bohr^3. In order to allow comparison of various systems this value should be kept fix (in critical cases, one may also check the FOD plot with a a smaller value of $\sigma = 0.002$ e/Bohr^3 for comparison). The FOD is strictly positive in all space and resembles orbital densities (e.g., π-shape in large polyenes) or the total charge density for an ideal 'metal' with complete orbital degeneracy in simple cases.

Basically, ρ^{FOD} can be plotted analogously to an electron density using the `orca_plot` utility program and the generated `basename.scfp_fod` density file that is stored in the `basename.scfp_fod` is stored in the `basename.densities` container.

To do so call `orca_plot` in interactive mode via

```
orca_plot basename.gbw -i
```

The interface will now show various plot options.

```
PlotType        ... MO-PLOT
MO/Operator     ... 0 0
Output file     ... (null)
Format          ... Grid3D/Binary
Resolution      ... 40 40 40
Boundaries      ...   -22.359392    16.311484 (x direction)
                      -12.628859    13.574504 (y direction)
                       -7.001812     7.002006 (z direction)

       1 - Enter type of plot
       2 - Enter no of orbital to plot
       3 - Enter operator of orbital (0=alpha,1=beta)
       4 - Enter number of grid intervals
       5 - Select output file format
       6 - Plot CIS/TD-DFT difference densities
       7 - Plot CIS/TD-DFT transition densities
       8 - Set AO(=1) vs MO(=0) to plot
       9 - List all available densities
      10 - Perform Density Algebraic Operations

      11 - Generate the plot
      12 - exit this program
Enter a number:
```

To check all available densities in the `basename.densities` container one can use the `9 - List all available densities` option.

```
---------------------
List of density names
---------------------

Index:                                Name of Density
-----------------------------------------------------
    0:                              basename.scfp_fod   <--- required for FOD plot
    1:                                  basename.scfp
    2:                                  basename.scfr
```

The most general way to create the FOD plot is now to create a `.cube` file that can be visualized with many external programs like ChimeraX or Chemcraft (cf. *Graphical User Interfaces*).

To do so we use the following subsequent user inputs to `orca_plot`:

```
1 (type of plot)
2 (electron density)
n (default name: no)
basename.scfp_fod (name of the FOD file)
4 (number of grid intervals)
120 (NGrid)
5 (output file format)
7 (cube)
10 (generate plot)
11 (exit)
```

Warning

Note that producing `.cube` files can become very large and their generation with `orca_plot` may take a considerable amount of time for larger molecules. This is particularly the case if high quality plots using tight

grid resolution settings (i.e., 120x120x120 resolution) are used for publication purposes.

It is also possible to generate `*.cube` files from ρ^{FOD} (analogously to electron density plots) with other programs that can read ORCA's `basename.gbw` and electron density files by simply using the `basename.scfp_fod` file instead of the `basename.scfp` file.

An example FOD plot for Ni(bis-dithiolene) is shown below (Fig. 5.2). It was generated with ChimeraX and a `.cube` file generated by `orca_plot` and clearly shows the strong FOD on the metal center and the adjacent ligands. This observation is in line with various studies on the multi-reference character of this complex.

Fig. 5.2: FOD plot at $\sigma = 0.005$ e/Bohr^3 (TPSS/def2-TZVP (T = 5000 K) level) for the NiBDT complex (FOD depicted in yellow).

More examples of FOD plots generated with the same setup can be found in the original publication and corresponding supplementary information.[588]

5.3.3 Example: *p*-Benzyne

In this example, the FOD analysis for the ground state of *p*-Benzyne is performed.

```
! FOD

* xyz 0 1
C 0.0000000  1.2077612  0.7161013
C 0.0000000  0.0000000  1.3596219
C 0.0000000 -1.2077612  0.7161013
C 0.0000000 -1.2077612 -0.7161013
C 0.0000000  0.0000000 -1.3596219
C 0.0000000  1.2077612 -0.7161013
H 0.0000000  2.1606260  1.2276695
H 0.0000000 -2.1606260  1.2276695
H 0.0000000 -2.1606260 -1.2276695
H 0.0000000  2.1606260 -1.2276695
*
```

The respective output reads:

```
-------------------------------------------------------------------------------
↪--------
                                    ORCA LEAN-SCF
                              memory conserving SCF solver
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------------------------------------------
↪--------

---------------------------------------D-I-I-S---------------------------------
↪--------
Iteration    Energy (Eh)           Delta-E    RMSDP     MaxDP     DIISErr   Damp ␣
↪Time(sec)
-------------------------------------------------------------------------------
↪--------
               ***  Starting incremental Fock matrix formation  ***
    1    -230.8982516003082139     0.00e+00  5.01e-03  1.01e-01  1.12e-01  0.700  ␣
↪1.7
Warning: op=0 Small HOMO/LUMO gap (  -0.021) - skipping pre-diagonalization
         Will do a full diagonalization
    2    -230.9463607195993120    -4.81e-02  1.15e-03  2.59e-02  4.06e-02  0.700  ␣
↪1.6
                              ***Turning on AO-DIIS***
  ... etc.
   12    -231.0033984839932089    -5.02e-09  3.33e-07  7.37e-06  7.95e-06  0.000  ␣
↪1.1
                 **** Energy Check signals convergence ****

FOD:
 Fermi smearing:E(HOMO(Eh)) = -0.201252 MUE = -0.179318 gap=   1.119 eV

 N_FOD =  0.920364
```

The high N_{FOD} already indicates strong SEC and checking the Mulliken reduced orbital charges based on $\rho^{FOD}(r)$ (see *Mulliken Population Analysis*) gives a first impression about the localization of hot electrons in the molecule. The printout for the first carbon atom is given below:

```
-----------------------------------------
FOD BASED MULLIKEN REDUCED ORBITAL CHARGES
-----------------------------------------
  0 C s       :     0.006371  s :     0.006371
      pz      :     0.016375  p :     0.030785
      px      :     0.009893
      py      :     0.004516
      dz2     :     0.004248  d :     0.010308
      dxz     :     0.000254
      dyz     :     0.004855
      dx2y2   :     0.000860
      dxy     :     0.000091
      f0      :     0.000006  f :     0.000378
      f+1     :     0.000014
      f-1     :     0.000309
      f+2     :     0.000002
      f-2     :     0.000006
      f+3     :     0.000010
      f-3     :     0.000032
```

If other population analysis printouts are wanted the user is referred to the Löwdin analysis (*Löwdin Population Analysis*) which is turned on by default using the total SCF density of the calculation, also in the case of finite electronic temperature.

The plot the ρ^{FOD} for *p*-Benzyne clearly shows the significant and rather delocalized FOD (^1A_g), thus indicating that multi-reference methods would be needed for reliable computational study of this molecule.

Fig. 5.3: FOD plot at $\sigma = 0.005$ e/Bohr^3 (TPSS/def2-TZVP (T = 5000 K) level) for the ^1A_g ground state of p-benzyne (FOD depicted in yellow).

5.4 Excited States Calculations

A plethora of methods to compute excited states exists in ORCA. Here we give a brief overview of the main methods, with focus on the basic usage and current capabilities of each. Detailed documentation for each method is indicated in each subsection.

Multi-reference methods, such as NEVPT2 or MRCI, are described elsewhere (*N-Electron Valence State Perturbation Theory (NEVPT2)*, *Multireference Configuration Interaction and Pertubation Theory (uncontracted)*) in the manual.

5.4.1 Excited States with RPA, CIS, TD-DFT and SF-TDA

ORCA features a module to perform TD-DFT, single-excitation CI (CIS) and RPA. The module works with either closed-shell (RHF or RKS) or unrestricted (UHF or UKS) reference wavefunctions. For DFT models the module automatically chooses TD-DFT and for HF wavefunctions the CIS model. If the RI approximation is used in the SCF part it will also be used in the excited states calculation. A detailed documentation is provided in section *Excited States via RPA, CIS, TD-DFT and SF-TDA*

General Use

In its simplest form it is only necessary to provide the number of roots sought:

```
! BP86 def2-SVP TightSCF

%tddft
  Nroots   10
  triplets true
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
end
```

Which gives the following output:

```
------------------------------------
TD-DFT/TDA EXCITED STATES (SINGLETS)
------------------------------------

the weight of the individual excitations are printed if larger than 1.0e-02

STATE  1:  E=   0.142238 au      3.870 eV    31217.5 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     7a ->   8a  :     0.999852 (c=  0.99992610)

STATE  2:  E=   0.279148 au      7.596 eV    61265.8 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     7a ->   9a  :     0.991058 (c=  0.99551880)

STATE  3:  E=   0.326547 au      8.886 eV    71668.9 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     5a ->   8a  :     0.992394 (c= -0.99618994)

STATE  4:  E=   0.339416 au      9.236 eV    74493.3 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     6a ->   8a  :     0.214928 (c=  0.46360308)
     7a ->  10a  :     0.760130 (c= -0.87185424)

STATE  5:  E=   0.357323 au      9.723 eV    78423.4 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     4a ->   8a  :     0.998607 (c=  0.99930350)

STATE  6:  E=   0.396031 au     10.777 eV    86918.7 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     7a ->  11a  :     0.995757 (c= -0.99787607)

STATE  7:  E=   0.412518 au     11.225 eV    90537.2 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     3a ->   8a  :     0.015703 (c=  0.12531336)
     6a ->   9a  :     0.982525 (c=  0.99122380)

STATE  8:  E=   0.420413 au     11.440 eV    92270.0 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     4a ->  10a  :     0.023644 (c= -0.15376603)
     5a ->   9a  :     0.184687 (c= -0.42975192)
     5a ->  11a  :     0.029413 (c= -0.17150093)
     6a ->   8a  :     0.535798 (c= -0.73198239)
     7a ->  10a  :     0.161330 (c= -0.40165919)
     7a ->  14a  :     0.031134 (c=  0.17644805)

STATE  9:  E=   0.454354 au     12.364 eV    99719.1 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     5a ->   9a  :     0.801630 (c=  0.89533782)
     5a ->  11a  :     0.012558 (c= -0.11206253)
     6a ->   8a  :     0.103311 (c= -0.32142015)
     7a ->  10a  :     0.051303 (c= -0.22650208)
     7a ->  14a  :     0.016544 (c=  0.12862428)

STATE 10:  E=   0.474384 au     12.909 eV   104115.2 cm**-1 <S**2> =   0.000000␣
↪Mult 1
     6a ->  10a  :     0.998860 (c= -0.99942977)
```

and the triplets:

```
------------------------------------
TD-DFT/TDA EXCITED STATES (TRIPLETS)
------------------------------------
```

(continues on next page)

(continued from previous page)

```
the weight of the individual excitations are printed if larger than 1.0e-02

STATE 11:  E=   0.114291 au      3.110 eV    25084.1 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     7a ->   8a  :     0.999453 (c=  0.99972624)

STATE 12:  E=   0.213324 au      5.805 eV    46819.1 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     6a ->   8a  :     0.996522 (c=  0.99825941)

STATE 13:  E=   0.255583 au      6.955 eV    56094.1 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     7a ->   9a  :     0.992767 (c=  0.99637714)

STATE 14:  E=   0.276345 au      7.520 eV    60650.8 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     5a ->   8a  :     0.998251 (c= -0.99912505)

STATE 15:  E=   0.316749 au      8.619 eV    69518.3 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     7a ->  10a  :     0.991502 (c= -0.99574190)

STATE 16:  E=   0.327793 au      8.920 eV    71942.2 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     4a ->   8a  :     0.994029 (c=  0.99701018)

STATE 17:  E=   0.377551 au     10.274 eV    82862.9 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     7a ->  11a  :     0.998586 (c= -0.99929259)

STATE 18:  E=   0.400159 au     10.889 eV    87824.7 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     3a ->   8a  :     0.062364 (c=  0.24972706)
     6a ->   9a  :     0.934672 (c= -0.96678411)

STATE 19:  E=   0.433339 au     11.792 eV    95107.0 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     5a ->   9a  :     0.988277 (c=  0.99412115)

STATE 20:  E=   0.445213 au     12.115 eV    97713.0 cm**-1 <S**2> =   2.000000␣
↪Mult 3
     3a ->   8a  :     0.934403 (c=  0.96664514)
     6a ->   9a  :     0.063400 (c=  0.25179341)
```

By default, it also prints the dipole absorption and circular dichroism spectra:

```
-----------------------------------------------------------------------------------
↪-----------------
                      ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)       (cm-1)     (nm)                 (au**2)    (au)    ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A  ->  1-3A    3.110029   25084.1   398.7   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  1-1A    3.870479   31217.5   320.3   0.000000000   0.00000  -0.00000  ␣
```

(continues on next page)

(continued from previous page)

```
↪0.00000   0.00000
  0-1A  ->  2-3A    5.804830   46819.1   213.6   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  3-3A    6.954778   56094.1   178.3   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  4-3A    7.519743   60650.8   164.9   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  2-1A    7.595990   61265.8   163.2   0.139374365   0.74893  -0.00000  ␣
↪0.86541  -0.00000
  0-1A  ->  5-3A    8.619170   69518.3   143.8   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  3-1A    8.885804   71668.9   139.5   0.002010847   0.00924   0.00000  -
↪0.00000  -0.09611
  0-1A  ->  6-3A    8.919695   71942.2   139.0   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  4-1A    9.235990   74493.3   134.2   0.021695302   0.09588  -0.30964  -
↪0.00000  -0.00000
  0-1A  ->  5-1A    9.723256   78423.4   127.5   0.000000000   0.00000  -0.00000  ␣
↪0.00000  -0.00000
  0-1A  ->  7-3A   10.273692   82862.9   120.7   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  6-1A   10.776542   86918.7   115.1   0.009793664   0.03709   0.00000  -
↪0.19260  -0.00000
  0-1A  ->  8-3A   10.888868   87824.7   113.9   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  7-1A   11.225180   90537.2   110.5   0.002033718   0.00740   0.00000  -
↪0.00000   0.08599
  0-1A  ->  8-1A   11.440020   92270.0   108.4   0.371220034   1.32448  -1.15086  -
↪0.00000   0.00000
  0-1A  ->  9-3A   11.791761   95107.0   105.1   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  -> 10-3A   12.114872   97713.0   102.3   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  9-1A   12.363588   99719.1   100.3   0.259360077   0.85625  -0.92534  ␣
↪0.00000   0.00000
  0-1A  -> 10-1A   12.908634  104115.2    96.0   0.000000000   0.00000  -0.00000  -
↪0.00000  -0.00000
```

```
-----------------------------------------------------------------------------------
↪-------
                     CD SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-------
     Transition      Energy     Energy  Wavelength    R        MX        MY       ␣
↪MZ
                      (eV)      (cm-1)     (nm)    (1e40*cgs)   (au)      (au)     ␣
↪(au)
-----------------------------------------------------------------------------------
↪-------
  0-1A  ->  1-3A    3.110029   25084.1   398.7   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  1-1A    3.870479   31217.5   320.3    0.00000   0.58235   0.00000   0.
↪00000
  0-1A  ->  2-3A    5.804830   46819.1   213.6   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  3-3A    6.954778   56094.1   178.3   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  4-3A    7.519743   60650.8   164.9   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  2-1A    7.595990   61265.8   163.2   -0.00000  -0.00000   0.00000   0.
↪32961
```

(continues on next page)

(continued from previous page)

```
  0-1A  ->  5-3A    8.619170   69518.3   143.8   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  3-1A    8.885804   71668.9   139.5   -0.00000  -0.00000  -0.73058   0.
↪00000
  0-1A  ->  6-3A    8.919695   71942.2   139.0   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  4-1A    9.235990   74493.3   134.2   -0.00000  -0.00000  -0.00000  -0.
↪00000
  0-1A  ->  5-1A    9.723256   78423.4   127.5    0.00000   0.31342  -0.00000   0.
↪00000
  0-1A  ->  7-3A   10.273692   82862.9   120.7   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  6-1A   10.776542   86918.7   115.1    0.00000   0.00000   0.00000   0.
↪58743
  0-1A  ->  8-3A   10.888868   87824.7   113.9   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  7-1A   11.225180   90537.2   110.5   -0.00000   0.00000  -0.06966   0.
↪00000
  0-1A  ->  8-1A   11.440020   92270.0   108.4   -0.00000  -0.00000  -0.00000   0.
↪00000
  0-1A  ->  9-3A   11.791761   95107.0   105.1   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  -> 10-3A   12.114872   97713.0   102.3   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  9-1A   12.363588   99719.1   100.3   -0.00000  -0.00000  -0.00000  -0.
↪00000
  0-1A  -> 10-1A   12.908634  104115.2    96.0    0.00000   0.00260   0.00000   0.
↪00000
```

Which can be processed with *orca_mapspc* for plotting.

The triplets parameter is only valid for closed-shell references. If chosen as true the program will also determine the triplet excitation energies in addition to the singlets.

The collinear spin-flip version of CIS/TDA (always starting from an open-shell reference!) can be invoked in a similar manner, using:

```
%tddft
  Nroots 5
  sf     true
end
```

Please check the section *Collinear Spin-Flip TDA (SF-TD-DFT)* for more details on how to use it, and how to understand its results.

If one wants to compute transient spectra, or transition dipoles starting from a given excited state, the option `DOTRANS` must be set to TRUE and an IROOT should be given for the initial state (the default is 1). If `DOTRANS ALL` is requested instead, the transition dipoles between all states are computed. The transient transition dipoles will then be printed after the normal spectra.

This option is currently only available for CIS/TDA and is done using the expectation value formalism, as the other transition dipole moments in ORCA.

```
%tddft
  IROOT    2
  DOTRANS  TRUE
           #or
  DOTRANS  ALL
end
```

Capabilities

Currently, the TD-DFT/CIS module is able to calculate excitation energies, absorption spectra and circular dichroism spectra. Within the TD-DFT method, magnetic circular dichroism (see *Simulation of (Magnetic) Circular Dichroism and Absorption Spectra*) and transient spectra can also be calculated.

Analytical gradients are available for TD-DFT in both restricted and unrestricted formalisms and also for the collinear spin-flip variant, which allows for geometry optimizations of excited states as described in *Excited State Geometry Optimization*.

5.4.2 Excited States with Restricted Open-shell CIS - ROCIS

In addition to the CIS/TD-DFT description of excited states, ORCA features the ROCIS method[589], [590], which performs configuration interaction with single excitations calculations using a restricted open-shell Hartree-Fock (ROHF) reference.

Starting from ORCA 6.0, the general-spin ROCIS (`GS-ROCIS`) [591] implementation is available. This new implementation can handle CSFs with arbitrary spin couplings obtained via the `CSF-ROHF` method as references.

The main scope of ROCIS is to calculate L-edge and M-edge X-ray absorption spectra (XAS) as well as X-ray magnetic circular dichroism (XMCD) and resonant inelastic X-ray scattering (RIXS). The computational costs are usually larger than TD-DFT, but significantly smaller than coupled-cluster based methods. Together with the pair natural orbital approach (`PNO-ROCIS`), spectra of medium to large molecular sizes are feasible to be calculated.

For a detailed documentation check *Excited States via ROCIS and ROCIS/DFT*.

General Use

The method is invoked by providing the number of roots sought in the `%rocis` block of the input file:

```
!def2-SVP TightSCF

%scf
 HFTyp ROHF
 ROHF_CASE HIGHSPIN
 ROHF_NEL[1] 2
end

%rocis
 NRoots 10
end

* xyz 0 3
 C 0  0       0.1058
 H 0  0.9910 -0.3174
 H 0 -0.9910 -0.3174
end
```

By default, the original `ROCIS` implementation is invoked, which is capable of dealing only with high-spin `ROHF` references, giving the following output:

```
Eigenvectors of ROCIS calculation:
the threshold for printing is: 1e-02

  i->a            single excitation from orbital i to a
  i->t->a         single excitation from orbital i to a with a spin flip in␣
↪orbital t
  i->t ; w->a     double excitation from orbital i to t and orbital w to a

STATE   0   Exc. Energy:   0.000mEh   0.000eV           0.0cm**-1
```

(continues on next page)

(continued from previous page)

```
        0                  :   0.9880  (0.993993)

STATE   1   Exc. Energy: 291.825mEh   7.941eV       64048.2cm**-1
       2->3                :   0.9602   (0.979900)

STATE   2   Exc. Energy: 307.258mEh   8.361eV       67435.4cm**-1
       1->4                :   0.0244   (-0.156226)
       4->5                :   0.9086   (-0.953183)
       4->11               :   0.0379   (-0.194741)
       1->3    ;    4->5     :   0.0126   (-0.112387)

STATE   3   Exc. Energy: 311.967mEh   8.489eV       68468.7cm**-1
       2->4                :   0.9558   (-0.977660)
       4->6                :   0.0181   (0.134462)

STATE   4   Exc. Energy: 349.147mEh   9.501eV       76629.0cm**-1
       3->5                :   0.8588   (-0.926723)
       3->11               :   0.0299   (-0.173056)
       1->3   ->5          :   0.0561   (0.236925)

STATE   5   Exc. Energy: 374.241mEh  10.184eV       82136.4cm**-1
       2->4                :   0.0187   (0.136885)
       4->6                :   0.9224   (0.960395)
       4->12               :   0.0360   (0.189800)

STATE   6   Exc. Energy: 413.285mEh  11.246eV       90705.6cm**-1
       3->6                :   0.8368   (0.914777)
       3->12               :   0.0307   (0.175082)
       1->6                :   0.0148   (-0.121572)
       1->3   ->6          :   0.0376   (-0.193912)
       2->3   ->5          :   0.0456   (0.213492)

STATE   7   Exc. Energy: 474.514mEh  12.912eV      104143.8cm**-1
       1->3                :   0.8308   (-0.911467)
       2->3   ->6          :   0.0826   (0.287351)
       2->3   ->12         :   0.0148   (0.121501)

STATE   8   Exc. Energy: 501.672mEh  13.651eV      110104.2cm**-1
       1->4                :   0.8364   (-0.914550)
       4->5                :   0.0249   (0.157804)
       4->7                :   0.0561   (0.236863)
       2->4    ;    3->6     :   0.0324   (-0.180124)

STATE   9   Exc. Energy: 511.571mEh  13.921eV      112276.9cm**-1
       3->6                :   0.0580   (0.240898)
       1->6                :   0.0166   (-0.128707)
       2->5                :   0.1178   (0.343223)
       2->3   ->5          :   0.3041   (-0.551423)
       2->4   ->5          :   0.2625   (-0.512374)
```

The general-spin version `GS-ROCIS` can be requested via:

```
!def2-SVP TightSCF

%scf
 HFTyp ROHF
 ROHF_CASE HIGHSPIN
 ROHF_NEL[1] 2
end

%rocis
```

(continues on next page)

(continued from previous page)

```
 DoGenROCIS true
 ReferenceMult 3
 NRoots 10
end

* xyz 0 3
 C 0  0       0.1058
 H 0  0.9910 -0.3174
 H 0 -0.9910 -0.3174
end
```

The output gives the resulting spin coupling in addition to orbital information:

```
Eigenvectors of ROCIS calculation:
the threshold for printing is: 1e-02

  i->a            single excitation from orbital i to a
  i->t ; w->a     double excitation from orbital i to t and orbital w to a

STATE   0   Exc. Energy:   0.000mEh   0.000eV           0.0cm**-1
        0                   :   0.9880   (-0.993993)  : spin coupling:  2+1+1 0

STATE   1   Exc. Energy: 291.825mEh   7.941eV       64048.2cm**-1
       2->3                 :   0.9602   (0.979900)  : spin coupling: +1 2+1 0

STATE   2   Exc. Energy: 307.258mEh   8.361eV       67435.4cm**-1
       1->4                 :   0.0244   (-0.156226) : spin coupling: +1+1 2 0
       4->5                 :   0.9086   (-0.953183) : spin coupling:  2+1 0+1
       4->11                :   0.0379   (-0.194741) : spin coupling:  2+1 0+1
       1->3    ;    4->5    :   0.0126   (0.112387)  : spin coupling: +1 2 0+1

STATE   3   Exc. Energy: 311.967mEh   8.489eV       68468.7cm**-1
       2->4                 :   0.9558   (0.977660)  : spin coupling: +1+1 2 0
       4->6                 :   0.0181   (-0.134462) : spin coupling:  2+1 0+1

STATE   4   Exc. Energy: 349.147mEh   9.501eV       76629.0cm**-1
       3->5                 :   0.8588   (-0.926723) : spin coupling:  2 0+1+1
       3->11                :   0.0299   (-0.173056) : spin coupling:  2 0+1+1
       1->5                 :   0.0695   (-0.263669) : spin coupling: +1-1+1+1

STATE   5   Exc. Energy: 374.241mEh  10.184eV       82136.4cm**-1
       2->4                 :   0.0187   (-0.136885) : spin coupling: +1+1 2 0
       4->6                 :   0.9224   (-0.960395) : spin coupling:  2+1 0+1
       4->12                :   0.0360   (-0.189800) : spin coupling:  2+1 0+1

STATE   6   Exc. Energy: 413.285mEh  11.246eV       90705.6cm**-1
       3->6                 :   0.8368   (-0.914777) : spin coupling:  2 0+1+1
       3->12                :   0.0307   (-0.175082) : spin coupling:  2 0+1+1
       1->6                 :   0.0609   (-0.246700) : spin coupling: +1-1+1+1
       2->5                 :   0.0242   (-0.155666) : spin coupling: +1+1+1-1
       2->5                 :   0.0345   (0.185684)  : spin coupling: +1-1+1+1

STATE   7   Exc. Energy: 474.514mEh  12.912eV      104143.8cm**-1
       1->3                 :   0.8308   (-0.911467) : spin coupling: +1 2+1 0
       2->6                 :   0.0374   (0.193474)  : spin coupling: +1+1+1-1
       2->6                 :   0.0631   (-0.251226) : spin coupling: +1-1+1+1
       2->12                :   0.0134   (-0.115919) : spin coupling: +1-1+1+1

STATE   8   Exc. Energy: 501.672mEh  13.651eV      110104.2cm**-1
       1->4                 :   0.8364   (0.914550)  : spin coupling: +1+1 2 0
       4->5                 :   0.0249   (-0.157804) : spin coupling:  2+1 0+1
```

(continues on next page)

(continued from previous page)

```
       4->7                  :   0.0561   (-0.236863) : spin coupling:  2+1 0+1
       2->4    ;    3->6     :   0.0324   (0.180124)  : spin coupling: +1 0 2+1

STATE   9   Exc. Energy: 511.571mEh  13.921eV      112276.9cm**-1
       3->6                  :   0.0580   (0.240898)  : spin coupling:  2 0+1+1
       1->6                  :   0.0311   (0.176432)  : spin coupling: +1+1+1-1
       2->5                  :   0.6111   (-0.781719) : spin coupling: +1+1+1-1
       2->5                  :   0.0557   (0.235994)  : spin coupling: +1+1-1+1
       2->5                  :   0.2060   (0.453844)  : spin coupling: +1-1+1+1
       2->11                 :   0.0115   (0.107060)  : spin coupling: +1-1+1+1
```

`GS-ROCIS` requires a valid `ROHF` solution as reference (either high-spin or a specific CSF). For this, one would use the `CSF-ROHF` method to obtain the reference wavefunction for which `GS-ROCIS` will be performed:

```
%scf
  HFTyp ROHF
  ROHF_CASE HIGHSPIN, USER_CSF or AF_CSF
end
```

For more details on the `CSF-ROHF` method, check *ROHF Options*.

The parametrized `ROCIS/DFT` formulation can be requested by:

```
%rocis
  DoGenROCIS            false   # ROCIS/DFT is available only for the high-spin␣
↪implementation of ROCIS.
  DoDFTCIS               true   # Switches on the ROCIS/DFT method.
  DFTCIS_c = 0.18, 0.20, 0.40   # Array input of the three parameters.
end
```

Important

- Currently, `ROCIS/DFT` is not implemented for the general-spin (`GS-ROCIS`) procedure.

Capabilities

At the present, ROCIS can be used to calculate excitation energies, absorption, circular dichroism and magnetic circular dichroism spectra. It is also capable of calculating resonant inelastic X-ray scattering (RIXS) spectra. Magnetic properties such as g-tensors, zero-field splittings, hyperfine couplings and electric field gradients are also available.

5.4.3 Excited States for Open-Shell Molecules with CASSCF Linear Response (MC-RPA)

ORCA has the possibility to calculate excitation energies, oscillator and rotatory strengths for CASSCF wave functions within the response theory (MC-RPA) formalism.[592, 593, 594] The main scope of MC-RPA is to simiulate UV/Vis and ECD absorption spectra of open-shell molecules like transition metal complexes and organic radicals. MC-RPA absorption spectra are usually more accurate than those obtained from the state-averaged CASSCF ansatz as orbital relaxation effects for excited states are taken into account. The computational costs are ususally larger than those of SA-CASSCF and should be comparable to a TD-DFT calculation for feasible active space sizes.

General Use

MC-RPA needs a converged state-specific CASSCF calculation of the electronic ground state. The only necessary information that the user has to provide is the desired number of excited states (roots). All other keywords are just needed to control the Davidson algorithm or post process the results. A minimal input for calculating the four lowest singlet excited states of ethylene could like the following:

```
#
# CASSCF + MCRPA for C2H4
#
! DEF2-SVP DEF2-TZVP/C VeryTightSCF

%casscf
 nel          2
 norb         2
 mult         1
 nroots       1
 gtol 1e-6
 etol 1e-10
end

%mcrpa
 nroots       8
end

* int 0 1
C  0  0  0 0         0    0
C  1  0  0 1.3385    0    0
H  1  2  0 1.07    120    0
H  1  2  3 1.07    120  180
H  2  1  3 1.07    120    0
H  2  1  3 1.07    120  180
*
```

After the residual norm is below a user-given threshold `TolR` we get the following information

```
 ------------------------------------------------------------------------------
↪----------

     All 8 RPA Roots CONVERGED Below 1.000e-05

 ------------------------------------------------------------------------------
↪----------
    3  (root    0)       3.352976e-01     3.489323e-07    Yes
       (root    1)       3.485288e-01     1.656998e-08    Yes
       (root    2)       3.514846e-01     2.178527e-08    Yes
       (root    3)       3.741213e-01     2.577113e-07    Yes
       (root    4)       3.743973e-01     2.416887e-08    Yes
       (root    5)       4.040700e-01     4.609207e-08    Yes
       (root    6)       4.479248e-01     1.240222e-08    Yes
       (root    7)       4.609744e-01     6.311327e-09    Yes
```

and the absorption and ECD spectrum

```
------------------------------------------------------------------------------
↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)       (cm-1)     (nm)                 (au**2)     (au)     ␣
```

(continues on next page)

(continued from previous page)

```
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A  ->  1-1A    9.123912   73589.3   135.9   0.430768702   1.92710   1.38820  -
↪0.00000  -0.00000
  0-1A  ->  2-1A    9.483952   76493.2   130.7   0.009915132   0.04267   0.00000  -
↪0.00000  -0.20657
  0-1A  ->  3-1A    9.564384   77142.0   129.6   0.000000000   0.00000   0.00000  ␣
↪0.00000  -0.00000
  0-1A  ->  4-1A   10.180358   82110.1   121.8   0.000000000   0.00000  -0.00000  -
↪0.00000  -0.00000
  0-1A  ->  5-1A   10.187869   82170.7   121.7   0.000000000   0.00000  -0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  6-1A   10.995304   88683.1   112.8   0.000000000   0.00000   0.00000  -
↪0.00000  -0.00000
  0-1A  ->  7-1A   12.188654   98308.1   101.7   0.000000000   0.00000  -0.00000  -
↪0.00000  -0.00000
  0-1A  ->  8-1A   12.543751  101172.2    98.8   0.000000000   0.00000  -0.00000  -
↪0.00000  -0.00000
...
-----------------------------------------------------------------------------------
↪-------
                     CD SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-------
     Transition      Energy     Energy  Wavelength    R        MX        MY       ␣
↪MZ
                      (eV)      (cm-1)     (nm)    (1e40*cgs)   (au)      (au)     ␣
↪(au)
-----------------------------------------------------------------------------------
↪-------
  0-1A  ->  1-1A    9.123912   73589.3   135.9   -0.00000   0.00000   0.00000   0.
↪00000
  0-1A  ->  2-1A    9.483952   76493.2   130.7   -0.00000  -0.00000   0.00000  -0.
↪00000
  0-1A  ->  3-1A    9.564384   77142.0   129.6   -0.00000   0.69943   0.00000   0.
↪00000
  0-1A  ->  4-1A   10.180358   82110.1   121.8   -0.00000  -0.15776   0.00000   0.
↪00000
  0-1A  ->  5-1A   10.187869   82170.7   121.7   -0.00000   0.00000   0.73302  -0.
↪00000
  0-1A  ->  6-1A   10.995304   88683.1   112.8    0.00000  -0.00000   0.54037   0.
↪00000
  0-1A  ->  7-1A   12.188654   98308.1   101.7    0.00000  -0.00000   0.00000   0.
↪00000
  0-1A  ->  8-1A   12.543751  101172.2    98.8   -0.00000  -0.00000   0.00000  -0.
↪90854
```

Capabilities

At the moment, we can simulate UV/Vis and ECD absorption spectra by computing excitation energies, oscillator and rotatory strengths (check section *One Photon Spectroscopy* for more information). The code is parallelized and the computational bottleneck is the integral direct AO-Fock matrix construction. All intermediates that depend on the number of states are stored on disk, which makes the MC-RPA implementation suitable for computing many low-lying electronic states of larger molecules. Abelian point-group symmetry can be exploited in the calculation (up to D_{2h}). But there are no calculations of spin-flip excitations possible at the moment. That means all excited states will have the same spin as the reference state, which is specified in the `%casscf` input block.

It is also possible to analyze and visualize the ground-to-excited-state transitions by means of natural transition orbitals[595] (NTO), which is explained in more detail in section *Excited States for Open-Shell Molecules with CASSCF*

Linear Response (MC-RPA).

For further details, please study our recent publications[594, 596].

5.4.4 Excited States with ADC2

Among the various approximate correlation methods available for excited states, one of the most popular one is algebraic diagrammatic construction(ADC) method. The ADC has it origin in the Green's function theory. It expands the energy and wave-function in perturbation order and can directly calculate the excitation energy, ionization potential and electron affinity, similar to that in the EOM-CCSD method. Because of the symmetric eigenvalue problem in ADC, the calculation of properties are more straight forward to calculate than EOM-CCSD. In ORCA, only the second-order approximation to ADC(ADC2) is implemented. It scales as O(N^5) power of the basis set.

General Use

The simplest way to perform an ADC2 calculation is via the usage of the `ADC2` keyword, together with the specification of the desired number of roots:

```
! ADC2 cc-pVDZ cc-pVDZ/C TightSCF

%mdci
  nroots 9
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The above input will call the ADC2 routine with default settings. The main output is a list of excitation energies, augmented with some further state specific data. The integral transformation in the ADC2 implementation of ORCA is done using the density-fitting approximation. Therefore, one need to specify an auxiliary basis. For the above input, the following output is obtained:

```
----------------------
ADC(2) RESULTS (RHS)
----------------------

IROOT=  1:  0.146914 au     3.998 eV   32243.8 cm**-1
  Amplitude    Excitation
  -0.116970     4 ->   8
   0.672069     7 ->   8
IROOT=  2:  0.286012 au     7.783 eV   62772.3 cm**-1
  Amplitude    Excitation
  -0.659777     7 ->   9
IROOT=  3:  0.341919 au     9.304 eV   75042.5 cm**-1
  Amplitude    Excitation
  -0.676913     5 ->   8
IROOT=  4:  0.352206 au     9.584 eV   77300.2 cm**-1
  Amplitude    Excitation
   0.126824     4 ->  10
   0.360690     6 ->   8
  -0.547670     7 ->  10
IROOT=  5:  0.393965 au    10.720 eV   86465.3 cm**-1
  Amplitude    Excitation
   0.551345     6 ->   8
   0.363450     7 ->  10
```

(continues on next page)

(continued from previous page)

```
  -0.109270      6 ->   8      6 ->   8
IROOT=  6:  0.404946 au    11.019 eV   88875.4 cm**-1
  Amplitude    Excitation
   0.669682      4 ->   8
   0.126557      7 ->   8
IROOT=  7:  0.412800 au    11.233 eV   90599.2 cm**-1
  Amplitude    Excitation
  -0.100274      4 ->  11
   0.671884      7 ->  11
IROOT=  8:  0.439251 au    11.953 eV   96404.5 cm**-1
  Amplitude    Excitation
  -0.674114      6 ->   9
   0.104541      6 ->   9      6 ->   8
IROOT=  9:  0.486582 au    13.241 eV  106792.5 cm**-1
  Amplitude    Excitation
  -0.654624      5 ->   9
```

The transition moment for ADC2 in ORCA is calculated using an EOM-like expectation value approach, unlike the traditionally used intermediate state representation. However, the two approaches gives almost identical result.

```
-------------------------------------------------------------------
                    SPECTRUM FOR LEFT-RIGHT TRANSITION MOMENTS
-------------------------------------------------------------------


-----------------------------------------------------------------------------------
↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)    (nm)                  (au**2)    (au)    ␣
↪ (au)      (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A  ->  1-1A    3.997726   32243.8   310.1   0.000000000   0.00000   0.00000  ␣
↪0.00000   0.00000
  0-1A  ->  2-1A    7.782776   62772.3   159.3   0.096710371   0.50720  -0.00000  -
↪0.70536   0.00000
  0-1A  ->  3-1A    9.304078   75042.5   133.3   0.002261744   0.00992  -0.00000  ␣
↪0.00000  -0.09835
  0-1A  ->  4-1A    9.584003   77300.2   129.4   0.007937829   0.03381  -0.18502  -
↪0.00000  -0.00000
  0-1A  ->  5-1A   10.720332   86465.3   115.7   0.465055079   1.77067   1.32377  ␣
↪0.00000  -0.00000
  0-1A  ->  6-1A   11.019150   88875.4   112.5   0.000000000   0.00000  -0.00000  ␣
↪0.00000  -0.00000
  0-1A  ->  7-1A   11.232869   90599.2   110.4   0.022236623   0.08080   0.00000  -
↪0.28105  -0.00000
  0-1A  ->  8-1A   11.952640   96404.5   103.7   0.009103120   0.03109   0.00000  -
↪0.00000   0.17328
  0-1A  ->  9-1A   13.240575  106792.5    93.6   0.071433742   0.22021  -0.46692  -
↪0.00000   0.00000
```

The IP and EA versions can be called using the keywords IP-ADC2 and EA-ADC2, respectively.

Capabilities

At present, the ADC2 module is able to perform excited, ionized and electron attached state calculations, only for closed-shell systems. No open-shell version of the ADC2 is currently available. Below are all the parameters that influence the ADC2 module.

```
%mdci
#ADC2 parameters - defaults displayed
  NDav 20               # maximum size of reduced space (i.e. 20*NRoots)
  CheckEachRoot true    # check convergence for each root separately
  RootHoming true       # apply root homing
  DoLanczos false       # use the Lanczos procedure rather than Davidson
  UseCISUpdate true     # use diagonal CIS for updating
  NInitS 0              # number of roots in the initial guess, if 0, use preset␣
↪value
  DoRootwise false      # solves for each root separately,
                        # more stable for large number of roots
  FOLLOWCIS false       # follows the initial singles guess
end
```

One can notice that features available in the ADC2 module is quite limited as compared to the EOM module and the option to specifically target the core-orbitals are yet not available. A word of caution, **The 'second order black magic' of ADC2 can fail in many of the cases.** The readers are encouraged to try the DLPNO based EOM-CCSD methods(*Excited States with DLPNO based coupled cluster methods*) which are much more accurate and computationally efficient.

5.4.5 Excited States with STEOM-CCSD

The STEOM-CCSD method provides an efficient way to calculate excitation energies, with an accuracy comparable to the EOM-CCSD approach, at a nominal cost. A detailed description will be given in Section *Excited States via STEOM-CCSD*.

General Use

The simplest way to perform a STEOM calculation is using the `STEOM-CCSD` keyword, together with the specification of the desired number of roots (`NRoots`):

```
! STEOM-CCSD cc-pVDZ TightSCF

%mdci
  NRoots 9        # Number of excited states
  DoDbfilter true # Remove doubly excited states
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The above input calls the STEOM routine with default settings, where, for instance, the doubly excited states are eliminated (`DoDbFilter true`). The main output is a list of excitation energies, augmented with some further state specific data. The STEOMCC approach in ORCA uses state-averaged CIS natural transition orbitals (NTO) for the selection of the active space. For the above input, the following output is obtained:

```
-------------------------------
STEOM-CCSD RESULTS (SINGLETS)
-------------------------------
```

(continues on next page)

(continued from previous page)

```
IROOT=  1:  0.145378 au     3.956 eV   31906.7 cm**-1
  Amplitude    Excitation
  -0.168322     4 ->   8
  -0.984801     7 ->   8
  Ground state amplitude: -0.000000

Percentage Active Character     99.91

  Amplitude    Excitation in Canonical Basis
  -0.166144     4 ->   8
  -0.975626     7 ->   8
   0.123172     7 ->  13

IROOT=  2:  0.309944 au     8.434 eV   68024.9 cm**-1
  Amplitude    Excitation
  -0.993139     7 ->   9
  Ground state amplitude:  0.000000

Percentage Active Character     99.94

  Amplitude    Excitation in Canonical Basis
  -0.989653     7 ->   9

IROOT=  3:  0.337588 au     9.186 eV   74092.0 cm**-1
  Amplitude    Excitation
  -0.993709     5 ->   8
  Ground state amplitude: -0.000002

Percentage Active Character     99.23

  Amplitude    Excitation in Canonical Basis
  -0.984171     5 ->   8
   0.136295     5 ->  13

IROOT=  4:  0.357393 au     9.725 eV   78438.7 cm**-1
  Amplitude    Excitation
  -0.186839     4 ->  10
  -0.755864     6 ->   8
  -0.601549     7 ->  10
  -0.112921     7 ->  12
  Ground state amplitude:  0.026385

Percentage Active Character     99.71

  Amplitude    Excitation in Canonical Basis
  -0.182608     4 ->  10
  -0.752095     6 ->   8
  -0.598472     7 ->  10

IROOT=  5:  0.386751 au    10.524 eV   84882.0 cm**-1
  Amplitude    Excitation
  -0.980511     4 ->   8
   0.178900     7 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.90

  Amplitude    Excitation in Canonical Basis
  -0.971593     4 ->   8
   0.121664     4 ->  13
```

(continues on next page)

(continued from previous page)

```
   0.179278     7 ->   8

IROOT=  6:  0.406225 au    11.054 eV   89156.2 cm**-1
  Amplitude    Excitation
   0.532674     6 ->   8
  -0.825021     7 ->  10
  Ground state amplitude: -0.065794

Percentage Active Character     99.70

  Amplitude    Excitation in Canonical Basis
   0.526200     6 ->   8
  -0.817279     7 ->  10

IROOT=  7:  0.421236 au    11.462 eV   92450.6 cm**-1
  Amplitude    Excitation
  -0.125749     4 ->  11
  -0.985406     7 ->  11
  Ground state amplitude:  0.000000

Percentage Active Character     99.85

  Amplitude    Excitation in Canonical Basis
   0.124222     4 ->  11
   0.983485     7 ->  11

IROOT=  8:  0.443588 au    12.071 eV   97356.3 cm**-1
  Amplitude    Excitation
   0.106457     3 ->   8
   0.992884     6 ->   9
  Ground state amplitude:  0.000092

Percentage Active Character     99.78

  Amplitude    Excitation in Canonical Basis
   0.106009     3 ->   8
   0.987228     6 ->   9

IROOT=  9:  0.512311 au    13.941 eV  112439.3 cm**-1
  Amplitude    Excitation
  -0.995561     6 ->  10
  Ground state amplitude: -0.000001

Percentage Active Character     99.94

  Amplitude    Excitation in Canonical Basis
  -0.985669     6 ->  10
   0.157781     6 ->  15
```

The first set of excitation amplitudes, printed for each root, have been calculated in the CIS NTO (Natural Transition Orbitals) basis. The second set of amplitudes have been evaluated in the RHF canonical basis.

Capabilities

At present, the STEOM routine is able to calculate excitation energies, for both closed- or open-shell systems, using an RHF or UHF reference function, respectively. It can be used for both serial and parallel calculations. The method is available in the back-tranformed PNO and DLPNO framework allowing the calculation of large molecules (Section *Capabilities* and *Excited States with DLPNO based coupled cluster methods*). In the closed-shell case (RHF), a lower scaling version can be invoked by setting the `CCSD2` keyword to true in the %mdci section, which sets a second order approximation to the exact parent approach. The transition moments can also be obtained for closed- and open-shell systems. For more details see Section *Excited States via STEOM-CCSD*.

5.4.6 Excited States with IH-FSMR-CCSD

The intermediate Hamiltonian Fock-space coupled cluster method (IH-FSMR-CCSD) provides an alternate way to calculate excitation energies, with an accuracy comparable to the STEOM-CCSD approach. A detailed description is given in Section *Excited States via IH-FSMR-CCSD*.

General Use

The IH-FSMR-CCSD calculation is called using the simple input keyword `IH-FSMR-CCSD` and specifying the desired number of excited states (`NRoots`) in the %mdci block.:

```
! IH-FSMR-CCSD cc-pVDZ TightSCF

%mdci
  nroots 6
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The above input will call the IH-FSMR-CCSD routine with default settings. The main output is a list of excitation energies, augmented with some further state specific data. The IH-FSMR-CCSD approach in ORCA uses state-averaged CIS natural transition orbitals(NTO) for the selection of the active space - similar to STEOM-CCSD. For the above input, the following output is obtained:

```
-------------------------------
IH-FSMR-CCSD RESULTS (SINGLETS)
-------------------------------

IROOT=  1:  0.144808 au     3.940 eV   31781.8 cm**-1
  Amplitude    Excitation
  -0.171178     4 ->   8
  -0.984024     7 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.96

  Amplitude    Excitation in Canonical Basis
  -0.169804     4 ->   8
  -0.976596     7 ->   8
   0.111105     7 ->  13

IROOT=  2:  0.309569 au     8.424 eV   67942.6 cm**-1
  Amplitude    Excitation
  -0.994029     7 ->   9
```

(continues on next page)

(continued from previous page)

```
  Ground state amplitude:  0.000000

Percentage Active Character     99.79

  Amplitude    Excitation in Canonical Basis
  -0.991036     7 ->   9

IROOT=  3:  0.337609 au     9.187 eV   74096.7 cm**-1
  Amplitude    Excitation
   0.992246     5 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.29

  Amplitude    Excitation in Canonical Basis
   0.985970     5 ->   8
  -0.120519     5 ->  13

IROOT=  4:  0.354726 au     9.653 eV   77853.3 cm**-1
  Amplitude    Excitation
  -0.167422     4 ->  10
   0.125754     5 ->  11
  -0.748485     6 ->   8
  -0.575997     7 ->  10
  -0.204999     7 ->  14
  Ground state amplitude:  0.000000

Percentage Active Character     94.11

Warning:: the state may have not converged with respect to active space
-------------------- Handle with Care --------------------

  Amplitude    Excitation in Canonical Basis
  -0.175024     4 ->  10
   0.110821     5 ->  11
  -0.745514     6 ->   8
  -0.609903     7 ->  10

IROOT=  5:  0.386134 au    10.507 eV   84746.6 cm**-1
  Amplitude    Excitation
  -0.980436     4 ->   8
   0.180972     7 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.91

  Amplitude    Excitation in Canonical Basis
  -0.972869     4 ->   8
   0.111877     4 ->  13
   0.180279     7 ->   8

IROOT=  6:  0.443256 au    12.062 eV   97283.4 cm**-1
  Amplitude    Excitation
  -0.110780     3 ->   8
  -0.991903     6 ->   9
  Ground state amplitude:  0.000000

Percentage Active Character     99.71

  Amplitude    Excitation in Canonical Basis
  -0.109728     3 ->   8
```

(continues on next page)

(continued from previous page)

```
  -0.988030     6 ->   9
```

The first set of excitation amplitudes, printed for each root, have been calculated in the CIS NTO (Natural Transition Orbitals) basis. The second set of amplitudes have been evaluated in the RHF canonical basis.

Capabilities

At present, the IH-FSMR-CCSD routine is able to calculate excitation energies, for only closed shell systems using an RHF reference. It can be used for both serial and parallel calculations. In the closed-shell case (RHF), a lower scaling version can be invoked by using bt-PNO approximation. The transition moments and solvation correction can be obtained using the CIS approximation.

5.4.7 Excited States with PNO based coupled cluster methods

The methods described in the previous section are performed over a canonical CCSD or MP2 ground state. The use of canonical CCSD amplitudes restricts the use of EOM-CC and STEOM-CC methods to small molecules. The use of MP2 amplitudes is possible (e.g. the EOM-CCSD(2) or STEOM-CCSD(2) approaches), but it seriously compromises the accuracy of the method.

The bt-PNO-EOM-CCSD methods gives an economical compromise between accuracy and computational cost by replacing the most expensive ground state CCSD calculation with a DLPNO based CCSD calculation. The typical deviation of the results from the canonical EOM-CCSD results is around 0.01 eV. A detailed description will be given in *Excited States via PNO-based coupled cluster*.

General Use

The simplest way to perform a PNO based EOM calculation is via the usage of the bt-PNO-EOM-CCSD keyword, together with the specification of the desired number of roots. The specification of an auxilary basis set is also required, just as for ground state DLPNO-CCSD calculations.

```
! bt-PNO-EOM-CCSD def2-TZVP def2-TZVP/C def2/J TightSCF

%mdci
  nroots 9
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The output is similar to that from a canonical EOM-CCSD calculation:

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1:  0.145420 au     3.957 eV   31915.9 cm**-1
  Amplitude    Excitation
   0.650351     7 ->   8
  -0.162540     7 ->  13
  Ground state amplitude:  0.000000
Percentage singles character=     92.33

IROOT=  2:  0.311168 au     8.467 eV   68293.4 cm**-1
```

(continues on next page)

(continued from previous page)

```
  Amplitude    Excitation
   0.650800     7 ->   9
  -0.155532     7 ->  11
  Ground state amplitude: -0.000000
Percentage singles character=     90.95

IROOT=  3:  0.337404 au     9.181 eV   74051.7 cm**-1
  Amplitude    Excitation
   0.652018     5 ->   8
  -0.169980     5 ->  13
  Ground state amplitude:  0.000000
Percentage singles character=     91.87

IROOT=  4:  0.348225 au     9.476 eV   76426.6 cm**-1
  Amplitude    Excitation
   0.152132     7 ->   9
   0.652819     7 ->  11
  Ground state amplitude:  0.000000
Percentage singles character=     92.35

IROOT=  5:  0.354668 au     9.651 eV   77840.6 cm**-1
  Amplitude    Excitation
   0.545649     6 ->   8
  -0.339835     7 ->  10
   0.170720     6 ->   8     6 ->   8
  Ground state amplitude:  0.032711
Percentage singles character=     87.10

IROOT=  6:  0.379606 au    10.330 eV   83313.9 cm**-1
  Amplitude    Excitation
   0.636153     4 ->   8
  -0.160301     4 ->  13
  -0.109552     7 ->   8
   0.143497     7 ->   8     6 ->   8
  Ground state amplitude:  0.000000
Percentage singles character=     88.55

IROOT=  7:  0.386807 au    10.526 eV   84894.3 cm**-1
  Amplitude    Excitation
   0.257812     6 ->   8
   0.584151     7 ->  10
   0.181783     7 ->  14
  Ground state amplitude:  0.038804
Percentage singles character=     90.30

IROOT=  8:  0.440552 au    11.988 eV   96690.1 cm**-1
  Amplitude    Excitation
  -0.655574     6 ->   9
  -0.104097     6 ->  16
  -0.112700     6 ->   9     6 ->   8
  Ground state amplitude:  0.000000
Percentage singles character=     87.92

IROOT=  9:  0.447219 au    12.169 eV   98153.2 cm**-1
  Amplitude    Excitation
   0.162756     7 ->   8
   0.651078     7 ->  13
  Ground state amplitude:  0.000000
Percentage singles character=     90.36
```

The IP and EA versions can be called by using the keywords bt-PNO-IP-EOM-CCSD and bt-PNO-EA-EOM-CCSD, respectively. Furthermore, the STEOM version can be invoked by using the keywords bt-PNO-STEOM-CCSD.

Capabilities

All of the features of canonical EOM-CC and STEOM-CC are available in the PNO based approaches for both closed- and open-shell systems.

5.4.8 Excited States with DLPNO based coupled cluster methods

The DLPNO-STEOM-CCSD method uses the full potential of DLPNO to reduce the computational scaling while keeping the accuracy of STEOM-CCSD.

Important

- DLPNO-STEOM-CCSD is currently only available for closed-shell systems!

General Use

The simplest way to perform a DLPNO based STEOM calculation is via the usage of the `STEOM-DLPNO-CCSD` keyword, together with the specification of the desired number of roots. The specification of an auxiliary basis set is also required, just as for ground state DLPNO-CCSD calculations.

As any CCSD methods, it is important to allow ORCA to access a significant amount of memory. In terms of scaling the limiting factor of the method is the size of temporary files and thus the disk space. For molecules above 1500 basis functions it starts to increase exponentially up to several teraoctets.

Here is the standard input we would recommend for STEOM-DLPNO-CCSD calculations. More information on the different keywords and other capabilities are available in the detailed part of the manual *Excited States via STEOM-CCSD*, *Excited States via DLPNO-STEOM-CCSD*. The following publications referenced some applications for this method either in organic molecules [597], [598] or for Semiconductors [599].

```
! STEOM-DLPNO-CCSD def2-TZVP def2-TZVP/C def2/J TightSCF

%mdci
  NRoots 6
  DoRootWise true
  OThresh 0.005
  VThresh 0.005
  TCutPNOSingles 1e-11
  NDAV 400
  DoStoreSTEOM true
  DoSimpleDens false
  AddL2Term True
  DTol 1e-5
end

* xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The output is similar to that from a canonical DLPNO-STEOM-CCSD calculation:

```
-------------------------------
STEOM-CCSD RESULTS (SINGLETS)
-------------------------------

IROOT=  1:  0.144275 au      3.926 eV   31664.7 cm**-1
```

(continues on next page)

(continued from previous page)

```
  Amplitude    Excitation
  -0.142146     4 ->   8
  -0.988793     7 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.79

  Amplitude    Excitation in Canonical Basis
  -0.134936     4 ->   8
  -0.955031     7 ->   8
   0.236743     7 ->  13

IROOT=  2:  0.308093 au     8.384 eV   67618.5 cm**-1
  Amplitude    Excitation
   0.971471     7 ->   9
   0.214898     7 ->  10
  Ground state amplitude: -0.000000

Percentage Active Character     99.67

  Amplitude    Excitation in Canonical Basis
   0.956929     7 ->   9
  -0.236568     7 ->  11
   0.102573     7 ->  16

IROOT=  3:  0.331790 au     9.028 eV   72819.4 cm**-1
  Amplitude    Excitation
   0.993677     5 ->   8
  Ground state amplitude: -0.000000

Percentage Active Character     98.87

  Amplitude    Excitation in Canonical Basis
   0.957221     5 ->   8
  -0.250140     5 ->  13
  -0.105951     5 ->  18

IROOT=  4:  0.346876 au     9.439 eV   76130.4 cm**-1
  Amplitude    Excitation
  -0.104901     4 ->  10
   0.198176     7 ->   9
  -0.972572     7 ->  10
  Ground state amplitude: -0.000000

Percentage Active Character     99.65

  Amplitude    Excitation in Canonical Basis
   0.100880     4 ->  11
   0.218873     7 ->   9
   0.956923     7 ->  11
  -0.113897     7 ->  19

IROOT=  5:  0.347460 au     9.455 eV   76258.7 cm**-1
  Amplitude    Excitation
   0.139551     4 ->  11
   0.106649     4 ->  12
  -0.801186     6 ->   8
   0.455613     7 ->  11
   0.302458     7 ->  12
  Ground state amplitude:  0.027275
```

(continues on next page)

(continued from previous page)

```
  Percentage Active Character     87.08

Warning:: the state may have not converged with respect to active space
-------------------- Handle with Care --------------------

  Amplitude    Excitation in Canonical Basis
   0.163791     4 ->  10
  -0.785701     6 ->   8
   0.159149     6 ->  13
   0.527833     7 ->  10
  -0.133085     7 ->  17

IROOT=  6:  0.379059 au    10.315 eV   83193.9 cm**-1
  Amplitude    Excitation
  -0.983700     4 ->   8
   0.155239     7 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     99.48

  Amplitude    Excitation in Canonical Basis
  -0.951092     4 ->   8
   0.235046     4 ->  13
   0.157714     7 ->   8
```

The IP and EA versions can be called by using the keywords IP-EOM-DLPNO-CCSD and EA-EOM-DLPNO-CCSD, respectively. As in canonical STEOM-CCSD, the first set of excitation amplitudes, printed for each root, are calculated in the CIS NTO (Natural Transition Orbitals) basis, while the second set is evaluated in the RHF canonical basis.

5.5 Excited State Dynamics

ORCA can now also be used to compute dynamic properties involving excited states such as absorption spectra, fluorescence and phosphorescence rates and spectra, as well as resonant Raman spectra using the new `ORCA_ESD` module. We do this by analytically solving the Fermi's Golden Rule-like equation from Quantum Electrodynamics (see the section *Excited State Dynamics*), using a path integral approach to the dynamics, as described in our recent papers [600, 601]. The computation of these rates relies on the harmonic approximation for the nuclear normal modes. Provided this approximation holds, the results closely match experimental data.

The theory can do most of what `ORCA_ASA` can and more, such as including vibronic coupling in forbidden transitions (the so-called Herzberg-Teller effect, HT), considering Duschinsky rotations between modes of different states, solving the equations using different coordinate systems, etc. There are also seven new approaches to obtain the excited state geometry and Hessian without necessarily optimizing its geometry. Many keywords and options are available, but most of the defaults already give good results. Let's get into specific examples, starting with the absorption spectrum. Please refer to section *Excited State Dynamics* for a complete keyword list and details.

5.5.1 Absorption Spectrum

The ideal model, Adiabatic Hessian (AH)

To predict absorption or emission rates, including all vibronic transitions, ideally, one requires both the ground state (GS) and excited state (ES) geometries and Hessians. For instance, when predicting the absorption spectrum for benzene, which exhibits one band above 220 nm corresponding to a symmetry-forbidden excitation to the S1 state, the process is straightforward. Ground state information can be obtained from (Sec. *Geometry Optimizations*):

```
!B3LYP DEF2-SVP OPT FREQ
* XYZFILE 0 1 BEN.xyz
```

and the S1 ES from (Sec. *Excited State Geometry Optimization*):

```
!B3LYP DEF2-SVP OPT FREQ
%TDDFT
  NROOTS   5
  IROOT    1
END
* XYZFILE 0 1 BEN_S1.xyz
```

Assuming DFT/TD-DFT here, but other methods can also be used (see *Tips, Tricks and Troubleshooting*). With both Hessians available, the ESD module can be accessed from:

```
!B3LYP DEF2-SVP TIGHTSCF ESD(ABS)
%TDDFT
  NROOTS     5
  IROOT      1
END
%ESD
  GSHESSIAN  "BEN.hess"
  ESHESSIAN  "BEN_S1.hess"
  DOHT       TRUE
END
* XYZFILE 0 1 BEN.xyz
```

Important

The geometry must match that in the GS Hessian when calling the ESD module. You can obtain it from the .xyz file after geometry optimization or directly copy it from the .hess file (remember to use BOHRS on the input to correct the units, if obtained from the .hess).

You must provide both names for the Hessians and set DOHT to TRUE here because the first transition of benzene is symmetry forbidden, with an oscillator strength of 2e-6. Therefore, all intensity arises from vibronic coupling (HT effect) [600]. In molecules with strongly allowed transitions, this parameter can typically remain FALSE (the default). Some calculation details are printed, including the computation of transition dipole derivatives for the HT component, and the spectrum is saved as BASENAME.spectrum.

```
Energy                  TotalSpectrum           IntensityFC             IntensityHT
10807.078728            2.545915e-02            2.067393e-07            2.545894e-
↪02
10828.022679            2.550974e-02            2.071508e-07            2.550954e-
↪02
10848.966630            2.556034e-02            2.075624e-07            2.556013e-
↪02
...
```

The first column has the total spectrum, but the contributions from the Franck-Condon part and the Herzberg-Teller part are also discriminated. As you can see, the FC intensity is less than 1% of the HT intensity here, highlighting the importance of including the HT effect. It is important to note that, in theory, the absorbance intensity values correspond to the experimental ε (in L mol cm^{-1}), and they depend on the spectral lineshape. The TotalSpectrum column can be plotted using any software to obtain the spectrum named Full AH spectrum (shown in blue), in Fig. 5.4 below.

Fig. 5.4: Here is the experimental absorption spectrum for benzene (shown in black on the left), alongside predictions made using ORCA_ESD at various PES approximations.

The spectrum obtained is very close to the experimental results at 298K, even when simply using all the defaults, and it could be further improved by adjusting parameters such as lineshape, as discussed in detail in Sec. *Fluorescence Rates and Spectrum* and Sec. *Excited State Dynamics*.

Note

- The path integral approach in ORCA_ESD is much faster than the more traditional approach of calculating all vibronic transitions with non-negligible intensities, one by one [602]. This is especially true for large systems, where the number of bright vibronic transitions may potentially scale exponentially, but our approach's scaling remains polynomial (in fact near linear scaling in favorable cases [600]). The price to pay is that one can no longer read off the compositions of the vibronic states from the spectrum, in other words, one cannot assign the peaks without doing further calculations. However, one can know whether a given vibrational mode contributes to a given peak, by repeating the ESD calculation with a few modes removed, and see if the peak is still present. This can be conveniently done using the MODELIST or SINGLEMODE keywords. More information can be found in Sec. *Excited State Dynamics*.
- The Huang-Rhys factors are important tools for qualitative and quantitative analysis of the contributions of each vibrational mode to the vibrationally resolved spectrum (and also to the transition rate constants, as will be discussed later). They can be requested by setting PRINTLEVEL in the %ESD module to 3 or above.

Of course, it is not always possible to obtain the excited state (ES) geometry due to root flipping, or it might be too costly for larger systems. Therefore, some approximations to the ES Potential Energy Surface (PES) have been developed.

The simplest model, Vertical Gradient (VG)

The minimal approximation, known as Vertical Gradient (VG), assumes that the excited state (ES) Hessian equals the ground state (GS) Hessian and extrapolates the ES geometry from the ES gradient and that Hessian using some step (Quasi-Newton or Augmented Hessian, which is the default here). Additionally, in this scenario, the simplest Displaced Oscillator (DO) model is employed, ensuring fast computation [600]. To use this level of approximation, simply provide an input like:

```
!B3LYP DEF2-SVP TIGHTSCF ESD(ABS)
%TDDFT
  NROOTS     5
  IROOT      1
END
%ESD
  GSHESSIAN  "BEN.hess"
  DOHT       TRUE
  HESSFLAG   VG    #DEFAULT
END
* XYZFILE 0 1 BEN.xyz
```

OBS: If no GSHESSIAN is given, it will automatically look for an BASENAME.hess file.

Choosing one of the methods in ORCA to compute excited state information is essential. Here, we utilize TD(A)-DFT with IROOT 1 to compute properties for the first excited state. TD(A)-DFT is currently the sole method offering analytic gradients for excited states; selecting any other method will automatically enforce NUMGRAD.

> **Important**
>
> Please note that certain methods, such as STEOM-DLPNO-CCSD, require significant time to compute numerical gradients. In such cases, we recommend using DFT/TD-DFT Hessians and employing the higher-level method solely for single points.

If everything is set correctly, after the regular single point calculation, the ESD module in ORCA will initiate. It proceeds to obtain the excited state (ES) geometry, compute derivatives, and predict the spectrum. The resulting normalized spectrum can be observed in Fig. 5.4, depicted in red. Due to such simple model, the spectrum is also simplified. While this simplicity is less critical for larger molecules, it highlight the potential benefit of employing an intermediate model.

A better model, Adiabatic Hessian After a Step (AHAS)

A reasonable compromise between a full geometry optimization and a simple step with the same Hessian is to perform a step and then recalculate the ES Hessian at that geometry. This approach is referred to here as Adiabatic Hessian After Step (AHAS). In our tests, it can be invoked with the following input:

```
!B3LYP DEF2-SVP TIGHTSCF ESD(ABS)
%TDDFT
  NROOTS     5
  IROOT      1
END
%ESD
  GSHESSIAN  "BEN.hess"
  DOHT       TRUE
  HESSFLAG   AHAS
END
* XYZFILE 0 1 BEN.xyz
```

The spectrum obtained corresponds to the green line in Fig. 5.4. As shown, it closely resembles the spectrum obtained using AH, where a full geometry optimization was performed. Although not set as the default, this method comes highly recommended based on our experience [600]. Another advantage of this approach is that the derivatives of the

transition dipole are computed simultaneously over Cartesian displacements on the ES structure using the numerical Hessian. Subsequently, these modes are straightforwardly converted.

OBS: The transition dipoles used in our formulation are always those of the FINAL state geometry. For absorption, this corresponds to the ES, so in AHAS, the derivatives are computed over this geometry. For fluorescence, the default behavior is to recompute the derivatives over the GS geometry. Alternatively, you can choose to save time and convert directly from ES to GS by setting CONVDER TRUE (though this is an approximation). For more details, refer to Sec. *Excited State Dynamics*.

Other PES options

There are also a few other options that can be set using HESSFLAG. For example, you can calculate the vertical ES Hessian over the GS geometry and perform a step, known as the **Vertical Hessian (HESSFLAG VH)** method. This method has the advantage that the geometry step is expected to be better because it does not assume the initial ES Hessian is equal to the GS Hessian. However, it is likely to encounter negative frequencies on that VH, since you are not at the ES minimum. By default, ORCA will turn negative frequencies into positive ones, issuing a warning if any were lower than -300 cm^{-1}. You can also choose to completely remove them (and the corresponding frequencies from the GS) by setting IFREQFLAG to REMOVE or leave them as negative with IFREQFLAG set to LEAVE under %ESD. Just be aware that an odd number of negative frequencies might disrupt the calculation of the correlation function, so be sure to check.

If your excited state is localized and you prefer not to recalculate the entire Hessian, you can opt for a **Hybrid Hessian (HH)** approach. This involves recomputing the ES Hessian only for specific atoms listed in HYBRID_HESS under %FREQ (*Vibrational Frequencies*). The HH method uses the GS Hessian as a base but adjusts it at the specified atoms. This computation can be performed either before or after the step, offering two variations: **Hybrid Hessian Before Step (HESSFLAG HHBS)** or **Hybrid Hessian After Step (HESSFLAG HHAS)**. When using either of these options, derivatives are recalculated across the modes as needed.

Another approach involves comparing the ES Hessian with the GS Hessian and selectively recomputing frequencies that differs. This method works by applying a displacement based on the GS Hessian and evaluating the resulting energy change. If the predicted mode matches the actual mode, the prediction should be accurate. However, if the difference exceeds a specified threshold, the gradient is computed, and the frequency for that mode is recalculated accordingly. The final ES Hessian is then derived from the **Updated Frequencies (UF)** and the original GS Hessian.

This approach offers the advantage of minimizing the computation of ES gradients typical in standard ES Hessians, thereby speeding up the process. By default, the system checks for frequency errors of approximately 20%. You can adjust this threshold using the UPDATEFREQERR flag; for instance, setting UPDATEFREQERR to 0.5 under %ESD allows for a larger error tolerance of 50%. Additionally, you can implement either the **Updated Frequencies Before Step (HESSFLAG UFBS)** or the **Updated Frequencies After Step (UFAS)** methods. Transition dipole derivatives are computed concurrently with the update process.

OBS: All these options apply to Fluorescence and resonant Raman as well.

Duschinsky rotations

The ES modes can sometimes be expressed as linear combinations of the GS modes (see Sec. *General Aspects of the Theory*), a phenomenon known in the literature as Duschinsky rotation [603]. In our formulation within ORCA_ESD, it is possible to account for this effect, which reflects a closer approximation to real-world scenarios, albeit at a higher computational cost.

You can enable this feature by setting USEJ TRUE; otherwise, the rotation matrix J defaults to unity. For instance, in the case of benzene, while the effect may not be pronounced, there is noticeable improvement in matching peak ratios with experimental data when incorporating rotations. Exploring this option may reveal more significant impacts in other cases.

Fig. 5.5: Experimental absorption spectrum for benzene (black on the left) and the effect of Duschinsky rotation on the spectrum.

Temperature effects

In our model, the effects of the Boltzmann distribution due to temperature are exactly accounted for [600]. The default temperature is set to 298.15 K, but you can specify any other temperature by adjusting the TEMP parameter under %ESD. However, it is important to note that when approaching temperatures close to 0 K, numerical issues may arise. For instance, if you encounter difficulties modeling a spectrum at 5 K or wish to predict a jet-cooled spectrum, setting TEMP to 0 will activate a set of equations specifically tailored for T=0 K conditions.

As can be seen in Fig. 5.6, at 0 K there are no hot bands and fewer peaks, while at 600 K there are many more possible transitions due to the population distribution over the GS.

Fig. 5.6: Predicted absorption spectrum for benzene at different temperatures.

Multistate Spectrum

If you want to predict a spectrum that includes many different states, you should ignore the IROOT flag in all modules and instead use the STATES flag under %ESD. For example, to predict the absorption spectra of pyrene in the gas phase and consider the first twenty states, you would specify:

```
!B3LYP DEF2-TZVP(-F) TIGHTSCF ESD(ABS)
%TDDFT
  NROOTS     20
END
%ESD
  GSHESSIAN  "PYR.hess"
  ESHESSIAN  "PYR_S1.hess"
  DOHT       TRUE
  STATES     1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
  UNIT       NM
END
* XYZFILE 0 1 PYR.xyz
```

This input would result in the spectra shown in Fig. 5.7. In this case, each individual spectrum for every state will be saved as BASENAME.spectrum.root1, BASENAME.spectrum.root2, etc., while the combined spectrum, which is the sum of all individual spectra, will be saved as BASENAME.spectrum.

Fig. 5.7: Predicted absorption spectrum for pyrene in gas phase (solid blue) in comparison to the experiment (dashed grey) at 298 K.

OBS: The flag UNIT can be used to control the output unit of the X axis. Its values can be CM-1, NM or EV and it only affects the OUTPUT, the INPUT should always be in cm^{-1}

5.5.2 Fluorescence Rates and Spectrum

General Aspects

The prediction of fluorescence rates and spectra can be performed in a manner analogous to absorption, as described above, by using ESD(FLUOR) on the main input line. You can select any of the methods described earlier to obtain the Potential Energy Surface (PES) by setting the appropriate HESSFLAG. The primary distinction is that the transition dipoles must correspond to the geometry of the ground state (GS), while all other aspects remain largely unchanged.

Fig. 5.8: Predicted absorption (right) and emission (left) spectrum for benzene in hexane at 298.15 K.

As depicted in Fig. 5.8, the fluorescence spectrum also closely matches the experimental data [600]. The difference observed in the absorption spectrum in Fig. 5.8, compared to previous spectra, arises because the experiment was conducted in a solvent environment. Therefore, we adjusted the linewidth to align with the experimental data.

OBS: It is common for the experimental lineshape to vary depending on the setup, and this can be adjusted using the LINEW flag (in cm^{-1}). There are four options for the lineshape function controlled by the LINES flag: DELTA (for a Dirac delta function), LORENTZ (default), GAUSS (for a Gaussian function), and VOIGT (a Voigt profile, which is a product of Gaussian and Lorentzian functions).

OBS2: The DELE and TDIP keywords can be used to input adiabatic (not vertical!) excitation energy and transition dipole moment computed at a higher level of theory. This enables calculating the computationally intensive Hessians (especially the excited state Hessian) at a low level of theory without compromising the accuracy. For more details, see Sec. *Mixing methods*.

If you need to control the lineshapes separately for Gaussian (GAUSS) and Lorentzian (LORENTZ), you can set LINEW for Lorentzian and INLINEW for Gaussian (where "I" stands for Inhomogeneous Line Width).

```
!B3LYP DEF2-SVP TIGHTSCF ESD(FLUOR)
%TDDFT
```

(continues on next page)

(continued from previous page)

```
  NROOTS     5
  IROOT      1
END
%ESD
  GSHESSIAN  "BEN.hess"
  ESHESSIAN  "BEN_S1.hess"
  DOHT       TRUE
  LINES      VOIGT
  LINEW      75
  INLINEW    200
END
* XYZFILE 0 1 BEN.xyz
```

OBS: The LINEW and INLINEW are NOT the full width half maximum ($FWHM$) of these curves. However, they are related to them by: $FWHM_{lorentz} = 2 \times LINEW$; $FWHM_{gauss} = 2.355 \times INLINEW$.

For the VOIGT curve, it is a little more complicated but in terms of the other FWHMs, it can be approximated as: $FWHM_{voigt} = 0.5346 \times FWHM_{lorentz} + \sqrt{(0.2166 \times FWHM^2_{lorentz} + FWHM^2_{gauss})}$.

Rates and Examples

When you select ESD(FLUOR) on the main input, the fluorescence rate will be printed at the end of the output, with contributions from Franck-Condon (FC) and Herzberg-Teller (HT) mechanisms discriminated. If you use CPCM, the rate will be multiplied by the square of the refractive index, following Strickler and Berg [604].

If you calculate a rate without CPCM but still want to account for the solvent effect, remember to multiply the final rate by this factor. Below is an excerpt from the output of a calculation with CPCM (hexane):

> **Warning**
>
> Whenever using ESD with CIS/TD-DFT and solvation, CPCMEQ will be set to TRUE by default, since the excited state should be under equilibrium conditions! More info in *Including solvation effects via LR-CPCM theory*.

```
...
          ***Everything is set, now computing the correlation function***

Homogeneous linewidht is:                        50.00 cm-1
Number of points:                                131072
Maximum time:                                        1592.65 fs
Spectral resolution:                               3.33 cm-1
Temperature used:                                298.15 K
Z value:                                        5.099843e-42
Energy difference:                                41049.37 cm-1
Reference transition dipole (x,y,z):                (0.00004 0.00000),
                                                    (0.00002 0.00000),
                                                    (-0.00058 0.00000)
Calculating correlation function:                ...done
Last element of the correlation function:        0.000000,-0.000000
Computing the Fourier Transform:                ...done

The calculated fluorescence rate constant is        1.688355e+06 s-1*
with 0.00% from FC and 100.00% from HT

*The rate is multiplied by the square of the refractive index

The fluorescence spectrum was saved in                BASENAME.spectrum
```

In one of our theory papers, we investigated the calculation of fluorescence rates for the set of molecules presented in Fig. 5.9. The results are summarized in Fig. 5.10 for some of the methods used to obtain the Potential Energy Surface (PES) mentioned.

Fig. 5.9: The set of molecules studied, with rates on Fig. 5.10.

Fig. 5.10: Predicted emission rates for various molecules in hexane at 298.15 K. The numbers below the labels are the HT contribution to the rates.

5.5.3 Phosphorescence Rates and Spectrum

General Aspects

As with fluorescence, phosphorescence rates and spectra can be calculated if spin-orbit coupling is included in the excited state module (please refer to the relevant publication [601]). To enable this, ESD(PHOSP) must be selected in the main input, and both a GSHESSIAN and a TSHESSIAN must be provided. The triplet Hessian can be computed analytically from the spin-adapted triplets.

```
!B3LYP DEF2-TZVP(-F) CPCM(ETHANOL) OPT FREQ
%TDDFT
  NROOTS 5
  IROOTMULT TRIPLET
END
* XYZ 0 1
C          -0.82240        -0.05739         0.00515
C           0.42295         0.77803         0.02146
H          -0.85252        -0.69527         0.89195
H          -0.85090        -0.66429        -0.90325
H          -1.69889         0.59680         0.01431
C           1.74379         0.02561        -0.01818
C           2.98907         0.86121        -0.00686
H           3.01366         1.50199        -0.89176
H           3.86561         0.20724        -0.02398
H           3.02300         1.46514         0.90332
O           0.42398         2.00161         0.06749
O           1.74282        -1.19814        -0.05965
*
```

or, in this case, by computing the ground state triplet by simply setting the multiplicity to three:

```
!B3LYP DEF2-TZVP(-F) CPCM(ETHANOL) OPT FREQ
* XYZFILE 0 3 BIA.xyz
```

Alternatively, one can use methods like VG, AHAS, etc., to approximate the triplet geometry and Hessian. However, this approach requires preparing the Hessian in a separate ESD run (Sec. *Approximations to the excited state PES*).

Additionally, you must input the adiabatic energy difference between the ground singlet and ground triplet states at their respective geometries (without any zero-point energy correction) using the DELE flag under %ESD. In this case, the spin-adapted triplet computed previously serves as our reference triplet state. An example input for the rate calculation using TDDFT is as follows:

```
!B3LYP DEF2-TZVP(-F) TIGHTSCF CPCM(ETHANOL) ESD(PHOSP) RI-SOMF(1X)
%TDDFT
  NROOTS  20
  DOSOC   TRUE
  TDA     FALSE
  IROOT   1
END
%ESD    GSHESSIAN       "BIA.hess"
        TSHESSIAN       "BIA_T1.hess"
        DOHT            TRUE
        DELE            17260
END
* XYZFILE 0 1 BIA.xyz

$NEW_JOB

!B3LYP DEF2-TZVP(-F) TIGHTSCF CPCM(ETHANOL) ESD(PHOSP) RI-SOMF(1X)
%TDDFT
  NROOTS  20
  DOSOC   TRUE
  TDA     FALSE
  IROOT   2
END
%ESD    GSHESSIAN       "BIA.hess"
        TSHESSIAN       "BIA_T1.hess"
        DOHT            TRUE
        DELE            17260
END
* XYZFILE 0 1 BIA.xyz

$NEW_JOB

!B3LYP DEF2-TZVP(-F) TIGHTSCF CPCM(ETHANOL) ESD(PHOSP) RI-SOMF(1X)
%TDDFT
  NROOTS  20
  DOSOC   TRUE
  TDA     FALSE
  IROOT   3
END
%ESD    GSHESSIAN       "BIA.hess"
        TSHESSIAN       "BIA_T1.hess"
        DOHT            TRUE
        DELE            17260
END
* XYZFILE 0 1 BIA.xyz
```

Phosphorescence rate calculation are always accompanied by the generation of the vibrationally resolved phosphorescence spectrum, which can be visualized in the same way as fluorescence spectra.

OBS.: When computing phosphorescence rates, each rate from individual spin sub-levels must be requested separately. You may use the $NEW_JOB option, just changing the `IROOT`, to write everything in a single input. After SOC, the three triplet states (T_1 with M_S = -1, 0 and +1) will split into IROOTs 1, 2 and 3, and all of them must be included when computing the final phosphorescence rate. In this case, it is reasonable to assume that the geometries and Hessians of these spin sub-levels are the same, and we will use the same .hess file for all three.

OBS2.: The ground state geometry should be used in the input file, similar to the case of fluorescence (vide supra).

OBS3.: Apart from DELE, one can also use the TDIP keyword to input a high-level transition dipole moment, similar to the fluorescence case. This enables e.g. the calculation of phosphorescence rates/spectra using e.g. NEVPT2 or

DLPNO-STEOM-CCSD transition dipole moments, with (TD)DFT geometries and Hessians. Note however that the transition dipole moment in phosphorescence processes is complex, so 6 instead of 3 components are required.

Here, we are computing the rate and spectrum for biacetyl in ethanol at 298 K. The geometries and Hessians were obtained as previously described, with the ground triplet computed from a simple open-shell calculation. To compute the rate, the flag DOSOC must be set to TRUE under %TDDFT (Sec *Spin-orbit coupling*), or the respective module, and it is advisable to set a large number of roots to ensure a good mixing of states.

Please note that we have chosen the RI-SOMF(1X) option for the spin-orbit coupling integrals, but any of the available methods can be used (Sec. *The Spin-Orbit Coupling Operator*).

Calculation of rates

As you can see, the predicted spectra for biacetyl (Fig. 5.11) are quite close to the experimental results [601, 605]. The calculation of the phosphorescence rate is more complex because there are three triplet states that contribute. Therefore, the observed rate must be taken as an average of these three states.

$$k_{av}^{phosp} = \frac{k_1 + k_2 + k_3}{3}$$

To be even more strict and account for the Boltzmann population distribution at a given temperature T caused by the Zero Field Splitting (ZFS), one should use [606]:

$$k_{av}^{phosp} = \frac{k_1 + k_2 e^{-(\Delta E_{1,2}/k_B T)} + k_3 e^{-(\Delta E_{1,3}/k_B T)}}{1 + e^{-(\Delta E_{1,2}/k_B T)} + e^{-(\Delta E_{1,3}/k_B T)}} \tag{5.28}$$

where $\Delta E_{1,2}$ is the energy difference between the first and second states, and so on.

After completion of each calculation, the rates for the three triplets were 8.91 s^{-1}, 0.55 s^{-1}, and 284 s^{-1}. Using (5.28), the final calculated rate is about 98 s^{-1}, while the best experimental value is 102 s^{-1} (at 77K) [607], with about 40% deriving from the HT effect.

Fig. 5.11: The experimental (dashed red) and theoretical (solid black, displaced by about 2800 cm^{-1}) phosphorescence spectra for biacetyl, in ethanol at 298 K.

OBS: A subtlety arises when the final state is not a singlet state, for example in radical phosphorescence (doublet ground state) or singlet oxygen phosphorescence (triplet ground state). In this case the most rigorous treatment would be to sum over the final states but average over the initial states. For example, with quartet-to-doublet phosphorescence one gets

$$k_{av}^{phosp} = \frac{k_{1\to1} + k_{2\to1}e^{-(\Delta E_{1,2}/k_BT)} + k_{3\to1}e^{-(\Delta E_{1,3}/k_BT)} + k_{4\to1}e^{-(\Delta E_{1,4}/k_BT)}}{1 + e^{-(\Delta E_{1,2}/k_BT)} + e^{-(\Delta E_{1,3}/k_BT)} + e^{-(\Delta E_{1,4}/k_BT)}} + \frac{k_{1\to2} + k_{2\to2}e^{-(\Delta E_{1,2}/k_BT)} + k_{3\to2}e^{-(\Delta E_{1,3}/k_BT)} + k_{4\to2}e^{-(\Delta E_{1,4}/k_BT)}}{1 + e^{-(\Delta E_{1,2}/k_BT)} + e^{-(\Delta E_{1,3}/k_BT)} + e^{-(\Delta E_{1,4}/k_BT)}}$$

where $k_{2\to1}$ is the phosphorescence rate constant from the second sublevel of the initial quartet to the first sublevel of the final doublet, etc. Note that the Boltzmann factors of the final state do not enter the expression. Unfortunately, since U-TDDFT is spin contaminated and unsuitable for calculating SOC-corrected transition dipole moments, the transition dipoles in this case have to be calculated by more advanced methods, such as DFT/ROCIS or multireference methods. The transition dipole should then be given in the input file using the TDIP keyword.

5.5.4 Intersystem Crossing Rates (unpublished)

General Aspects

Yet another application of the path integral approach is to compute intersystem crossing rates, or non-radiative transition rates between states of different multiplicities. This can be calculated if one has two geometries, two Hessians, and the relevant spin-orbit coupling matrix elements.

The input is similar to those discussed above. Here ESD(ISC) should be used on the main input to indicate an InterSystem Crossing calculation, and the Hessians should be provided by ISCISHESSIAN and ISCFSHESSIAN for the initial and final states, respectively. Please note that the geometry used in the input file should correspond to that of the FINAL state, specified through the ISCFSHESSIAN flag. The relevant matrix elements can be computed using any method available in ORCA and inputted as SOCME Re,Im under %ESD where Re and Im represent its real and imaginary parts (**in atomic units!**).

As a simple example, one could compute the excited singlet and ground triplet geometries and Hessians for anthracene using TD-DFT. Then, compute the spin-orbit coupling (SOC) matrix elements for a specific triplet spin-sublevel using the same method (see the details below), potentially employing methods like CASSCF, MRCI, STEOM-CCSD, or another theoretical level. Finally, obtain the intersystem crossing (ISC) rates using an input such as:

```
!ESD(ISC) NOITER
%ESD
  ISCISHESSIAN  "ANT_S1.hess"
  ISCFSHESSIAN  "ANT_T1.hess"
  DELE          11548
  SOCME         0.0, 2.33e-5
END
* XYZFILE 0 1 ANT_T1.xyz
```

The SOCMEs between a singlet state and a triplet state consist of three complex numbers, not just one, because the triplet state has three sublevels. If the user uses the SOCME of one of the sublevels as input to the ISC rate calculation, this gives the ISC rate associated with that particular sublevel. However, experimentally one usually does not distinguish the three sublevels of a triplet state, and experimentally ISC rates are reported as if the three sublevels of a triplet state are the same species. Therefore, the rate of singlet-to-triplet ISC is the *sum* of the ISC rates from the singlet state to the three triplet sublevels. Fortunately, in case the Herzberg-Teller effect (vide infra) can be neglected, it is not necessary to perform three rate calculations and add up the rates, since the rate is proportional to the squared modulus of the SOCME. Thus, one can run a *single* ESD calculation where the SOCME is the square root of the summed squared moduli of the three SOCME components.

As an illustration, consider the S_1 to T_1 ISC rate of acetophenone. First, we optimize the S_0 geometry, and (after manually tweaking the geometry to break mirror symmetry) use it as an initial guess for the geometry optimization and frequency calculations of the S_1 and T_1 states. Then, we calculate the S_1-T_1 SOCMEs at the T_1 geometry (note that, as usual, final state geometries should be used for ESD calculations; this may differ from some programs other

than ORCA). These calculations are conveniently done using a compound script, although the individual steps can of course also be done using separate input files.

```
%pal nprocs 16 end

* xyz 0 1
  C      1.512698    7.783764   -0.013405
  C      2.900029    7.735359   -0.016012
  C      3.664170    8.950745    0.000408
  C      2.952045   10.199539    0.007630
  C      1.564497   10.214795    0.009890
  C      0.827311    9.015246    0.000726
  H      0.946857    6.847253   -0.027731
  H      3.394565    6.763402   -0.041976
  H      3.505090   11.140193    0.017889
  H      1.039592   11.175094    0.019479
  H     -0.265468    9.038211    0.000969
  C      5.059695    8.958412    0.000442
  O      5.733735   10.161120   -0.048672
  C      5.927824    7.713913    0.020029
  H      5.442615    6.938582    0.639989
  H      6.073177    7.326421   -1.014490
  H      6.923426    7.950459    0.447785
 *

%compound

new_step # Compound 1: S1 opt
! B3LYP def2-SV(P) opt tightopt freq

%tddft
tda false
nroots 2
iroot 1
end

step_end

new_step # Compound 2: T1 opt
! B3LYP def2-SV(P) opt tightopt freq

* xyz 0 3
  C      1.512698    7.783764   -0.013405
  C      2.900029    7.735359   -0.016012
  C      3.664170    8.950745    0.000408
  C      2.952045   10.199539    0.007630
  C      1.564497   10.214795    0.009890
  C      0.827311    9.015246    0.000726
  H      0.946857    6.847253   -0.027731
  H      3.394565    6.763402   -0.041976
  H      3.505090   11.140193    0.017889
  H      1.039592   11.175094    0.019479
  H     -0.265468    9.038211    0.000969
  C      5.059695    8.958412    0.000442
  O      5.733735   10.161120   -0.048672
  C      5.927824    7.713913    0.020029
  H      5.442615    6.938582    0.639989
  H      6.073177    7.326421   -1.014490
  H      6.923426    7.950459    0.447785
 *

step_end
```

(continues on next page)

(continued from previous page)

```
new_step # Compound 3: SOC at T1 geometry
! B3LYP def2-SV(P)

%tddft
tda false
nroots 3
triplets true
dosoc true
end

# Here it is assumed that the input file is named "a.inp"
* xyzfile 0 1 a_Compound_2.xyz

step_end
end
```

After verifying that neither of the Hessians have imaginary frequencies (which is very important!), we can read the computed SOCMEs:

```
      ------------------------------------------------------------------------
↪---
                        CALCULATED SOCME BETWEEN TRIPLETS AND SINGLETS
      ------------------------------------------------------------------------
↪---
          Root                            <T|HSO|S>  (Re, Im) cm-1
        T      S              Z                      X                      Y
      ------------------------------------------------------------------------
↪---
        1      0    (   0.00 ,   -2.00 )    (   0.00 ,   29.24 )    (  -0.00 ,  -
↪41.11 )
        1      1    (   0.00 ,   -0.59 )    (   0.00 ,    1.79 )    (  -0.00 ,   -
↪2.57 )
        1      2    (   0.00 ,    0.52 )    (   0.00 ,   -3.19 )    (  -0.00 ,  ␣
↪3.21 )
        1      3    (   0.00 ,   -2.39 )    (   0.00 ,   14.57 )    (  -0.00 ,  -
↪19.53 )
        2      0    (   0.00 ,   -0.11 )    (   0.00 ,    2.78 )    (  -0.00 ,   -
↪2.72 )
        2      1    (   0.00 ,    2.25 )    (   0.00 ,  -20.07 )    (  -0.00 ,  ␣
↪28.07 )
        2      2    (   0.00 ,    0.02 )    (   0.00 ,   -0.13 )    (  -0.00 ,  ␣
↪0.29 )
        2      3    (   0.00 ,   -0.17 )    (   0.00 ,    0.59 )    (  -0.00 ,   -
↪1.27 )
        3      0    (   0.00 ,   -0.01 )    (   0.00 ,    0.05 )    (  -0.00 ,   -
↪0.06 )
        3      1    (   0.00 ,   -0.01 )    (   0.00 ,    0.19 )    (  -0.00 ,   -
↪1.36 )
        3      2    (   0.00 ,   -0.01 )    (   0.00 ,   -0.05 )    (  -0.00 ,   -
↪0.14 )
        3      3    (   0.00 ,   -0.00 )    (   0.00 ,    0.14 )    (  -0.00 ,   -
↪0.16 )
```

The "total" SOCME between S_1 and T_1 is then calculated, from the line that begins with "1 1", as

$$\sqrt{0.00^2 + (-0.59)^2 + 0.00^2 + 1.79^2 + 0.00^2 + 2.57^2} cm^{-1} = 3.19 cm^{-1} = 1.45 \times 10^{-5} au$$

One should therefore write `socme 1.45e-5` in the `%esd` block in the subsequent ISC rate calculation.

Importantly, the above approach is only applicable to singlet-to-triplet ISC, but not to triplet-to-singlet ISC (including, but not limited to, $T_1 \rightarrow S_0$ and $T_1 \rightarrow S_1$ processes). In the latter case, assuming that the triplet sublevels are

degenerate and in rapid equilibrium, we obtain that the observed rate constant is the *average*, not the sum, of the rate constants of the three triplet sublevels, because each triplet sublevel has a Boltzmann weight of 1/3. Therefore, the "effective" SOCME that should be plugged into the ESD module to get the observed rate constant is (here the squared modulus $|SOCME_x|^2$ should be calculated as $Re(SOCME_x)^2 + Im(SOCME_x)^2$, etc.)

$$SOCME_{av} = \sqrt{\frac{|SOCME_x|^2 + |SOCME_y|^2 + |SOCME_z|^2}{3}} \tag{5.29}$$

i.e. a factor of $\sqrt{3}$ smaller than the singlet-to-triplet case. However, both of the two assumptions (degenerate triplet sublevels, and rapid equilibrium between sublevels) may fail under certain circumstances, which may contribute to the error of the predicted rate. Nevertheless, in many cases the present, simple approach still provides a rate with at least the correct order of magnitude.

OBS.: The adiabatic energy difference is NOT computed automatically for ESD(ISC), so you must provide it in the input. This is the energy of the initial state minus the energy of the final state, each evaluated at its respective geometry.

OBS2.: All the other options concerning changes of coordinate system, Duschinsky rotation, etc., are also available here.

OBS3.: For many molecules, the $S_1 \rightarrow T_1$ ISC process is not the dominant ISC pathway. This is because the excited state compositions of S_1 and T_1 are often similar, and therefore ISC transitions between them frequently do not satisfy the El-Sayed rule. Even if the only experimentally observed excited states are S_1 and T_1, it may still be that the initial ISC is dominated by $S_1 \rightarrow T_n(n > 1)$, followed by ultrafast $T_n \rightarrow T_1$ internal conversion.

OBS4.: Similarly, if the molecule starts at a high singlet state $S_n(n > 1)$, the dominant ISC pathway is not necessarily the direct ISC from S_n to one of the triplet states. Rather, it is possible (but not necessarily true) that S_n first decays to a lower singlet excited state before the ISC occurs.

OBS5.: If you calculate the DELE or SOCMEs at a higher level of theory and use it as an input for the ESD calculation, please make sure that you have chosen the same excited state (in terms of state composition, not energy ordering) in the Hessian and DELE/SOCME calculations. For example, suppose that you have obtained the geometry and Hessian of the T_2 state, but the T_2 state of the higher level of theory has a very different state composition than the T_2 state at the level of theory used in the Hessian calculation; rather, it is T_3 at the high level of theory that shares the same composition as the T_2 state at the lower level of theory. In this case, you should use the SOCME related to T_3 in the SOCME output file.

OBS6.: The ESD module does not require that the final state of the ISC process is energetically lower than the initial state. Therefore, reverse ISC (RISC) rates can be calculated in exactly the same way as ISC rates. Note however that (1) if you want to supply the adiabatic energy difference using the DELE keyword, the energy difference should be negative; and (2) whether one should sum or average the three components of the SOCMEs depend on whether the final state is a triplet state, not whether the lower state is a triplet state.

OBS7.: ISC transitions between states other than singlets and triplets (for example between a doublet state an a quartet state) can also be calculated, provided that the SOCMEs are calculated by a properly spin-adapted or multireference method, such as DFT/ROCIS or NEVPT2. The squared moduli of the sublevels' SOCMEs should be summed over all final state spin sublevels but averaged over all initial state spin sublevels, similar to the phosphorescence case (*Calculation of rates*).

ISC, TD-DFT and the HT effect

In the anthracene example above, the result is an ISC rate (k_{ISC}) smaller than $1s^{-1}$, which is quite different from the experimental value of $10^8 s^{-1}$ at $77K$ [607]. The reason for this discrepancy, in this particular case, is because the ISC occurs primarily due to the Herzberg-Teller effect, which must also be included. To achieve this, one needs to compute the derivatives of the SOCMEs over the normal modes, currently feasible only using CIS/TD-DFT.

When using the %CIS/TDDFT option, you can control the SROOT and TROOT flags to select the singlet and triplet states for which SOCMEs are computed, and the TROOTSSL flag to specify the triplet spin-sublevel (1, 0, or -1).

In practice, to obtain an ISC rate (k_{ISC}) close enough to experimental values, one would need to consider all possible transitions between the initial singlet and all available final states. For anthracene, these are predicted to be the ground triplet (T_1) and the first excited triplet (T_2), consistent with experimental observations [608], while the next triplet

(T_3) is energetically too high to be significant (Fig. 5.12 below). An example input used to calculate the k_{ISC} from S_1 to T_1 at $77K$ is:

```
!B3LYP DEF2-TZVP(-F) TIGHTSCF ESD(ISC)
%TDDFT  NROOTS  5
        SROOT   1
        TROOT   1
        TROOTSSL 0
        DOSOC   TRUE
END
%ESD    ISCISHESS       "ANT_S1.hess"
        ISCFSHESS       "ANT_T1.hess"
        USEJ            TRUE
        DOHT            TRUE
        TEMP            77
        DELE            11548
END
* XYZFILE 0 1 ANT_T1.xyz

$NEW_JOB

!B3LYP DEF2-TZVP(-F) TIGHTSCF ESD(ISC)
%TDDFT  NROOTS  5
        SROOT   1
        TROOT   1
        TROOTSSL -1
        DOSOC   TRUE
END
%ESD    ISCISHESS       "ANT_S1.hess"
        ISCFSHESS       "ANT_T1.hess"
        USEJ            TRUE
        DOHT            TRUE
        TEMP            77
        DELE            11548
END
* XYZFILE 0 1 ANT_T1.xyz

...
```

Fig. 5.12: Scheme for the calculation of the intersystem crossing in anthracene. The $k_{ISC}(i)$ between S_1 and each triplet is a sum of all transitions to the spin-sublevels and the actual observed k_{ISC}^{obs}, which consolidates these transitions. On the right, there is a diagram illustrating the distribution of excited states with $E(S_1) - E(T_n)$ depicted on the side. Since T_3 is energetically too high, intersystem crossing involving T_3 can be safely neglected.

Then, the derivatives of the SOCME are computed and the rates are printed at the end. By performing the same

calculations for the T_2 states and summing up these values, a predicted $k_{ISC}^{obs} = 1.17 \times 10^8 s^{-1}$ can be obtained, much closer to the experimental value, which is associated with a large error anyway.

OBS.: In cases where the SOCME are relatively large, e.g., SOCME $> 5cm^{-1}$, the Herzberg-Teller effect might be negligible, and a simple Franck-Condon calculation should yield good results. This is typically applicable to molecules with heavy atoms, where vibronic coupling is less significant.

OBS2.: Always consider that there are actually THREE triplet spin-sublevels, and transitions from the singlet to all of them should be included.

OBS3.: ISC rates are extremely sensitive to energy differences. Exercise caution when calculating them. If a more accurate excited state method is available, it should be considered for prediction.

OBS4.: The Herzberg-Teller effect is not yet implemented for ISC transitions between states that are not singlets and triplets.

5.5.5 Internal Conversion Rates (unpublished)

The ESD module can also calculate internal conversion (IC) rates from an excited state to the ground state at the TDDFT and TDA levels. Apart from the ground state and excited state Hessians, the only additional quantity that needs to be calculated is the nonadiabatic coupling matrix elements (NACMEs).

The input file is simple:

```
# S1-S0 IC rate of azulene
! B3LYP D3 def2-SVP ESD(IC) CPCM(methanol)

%tddft
tda false # Full TDDFT is recommended over TDA
nroots 3
iroot 1 # Change to 2 for S2-S0 IC rate, etc.
nacme true # Calculate the NACME between the iroot-th root with the ground state
etf true # Use electron translation factor (recommended)
end

%esd
gshessian "azulene_S0.hess" # Ground state Hessian (B3LYP-D3/def2-SVP)
eshessian "azulene_S1.hess" # Excited state Hessian (TD-B3LYP-D3/def2-SVP)
usej true # Use Duschinsky rotation (recommended)
end

# Ground state geometry (B3LYP-D3/def2-SVP)
* xyzfile 0 1 azulene_S0.xyz
```

Here the S_0 geometry, as well as the S_0 and S_1 Hessian files, were obtained at the B3LYP-D3/def2-SVP level of theory. Note that the NACME calculation uses full TDDFT and also includes the electron-translation factor (ETF), which are the recommended practices in general. The "iroot 1" specifies that the initial state is S_1; the final state is always S_0 and this cannot be changed.

The computed IC rate constant is given near the end of the output file:

```
          ***Everything is set, now computing the correlation function***

Homogeneous linewidth is:     50.00 cm-1
Inhomogeneous linewidth is:     250.00 cm-1
Number of points:       32768
Maximum time:         157.86 fs
Temperature used:       298.15 K
Z value:          4.924300e-66
Cutoff for the correlation function:    1.00e-12
Adiabatic energy difference:      16699.89 cm-1
0-0 energy difference:        16382.09 cm-1
```

(continues on next page)

(continued from previous page)

```
Calculating correlation function:   ...done
Last element of the correlation function: -0.000000,-0.000000

The calculated internal conversion rate constant is 3.823146e+08 s-1

Total run time: 0 hours 2 minutes 50 seconds

                      ****ORCA ESD FINISHED WITHOUT ERROR****
```

For more accurate results, one may add explicit solvation shells, since implicit solvation models only describe the electrostatic and dispersive effects of the solvent on the solute, but cannot provide the extra vibrational degrees of freedom that can help dissipating the excitation energy. Conversely, by using an implicit solvent one also misses the effect of the solvent viscosity on inhibiting the internal conversion, which is particularly important when one wants to compare against experiments conducted at low temperatures.

5.5.6 Resonant Raman Spectrum

General Aspects

Using a theoretical framework similar to what was published for Absorption and Fluorescence, we have also developed a method to compute resonant Raman spectra for molecules [609]. In this implementation, one can employ all the methods to obtain the excited state potential energy surfaces (PES) mentioned earlier using HESSFLAG, and include Duschinsky rotations and even consider the Herzberg-Teller effect on top of it. This calculation can be initiated by using ESD(RR) or ESD(RRAMAN) on the first input line. It is important to note that by default, we calculate the "Scattering Factor" or "Raman Activity," as described by D. A. Long [610] (see Sec. *Resonant Raman Spectrum* for more information).

When using this module, the laser energy can be controlled by the LASERE flag. If no laser energy is specified, the 0-0 energy difference is used by default. You can select multiple energies by using LASERE 10000, 15000, 20000, etc., and if multiple energies are specified, a series of files named BASENAME.spectrum.LASERE will be saved. Additionally, it is possible to specify several states of interest using the STATES flag, but not both simultaneously.

As an example, let's predict the resonant Raman spectrum of the phenoxyl radical. You need at least a ground state geometry and Hessian, and then you can initiate the ESD calculation using:

```
!PBE0 DEF2-SVP TIGHTSCF ESD(RR)
%TDDFT
  NROOTS     5
  IROOT      3
END
%ESD
  GSHESSIAN  "PHE.hess"
  LASERE     28468
END
* XYZFILE 0 2 PHE.xyz
```

Important

The LASERE used in the input is NOT necessarily the same as the experimental one. It should be proportional to the theoretical transition energy. For example, if the experimental 0-0 ΔE is 30000 cm^{-1} and the laser energy used is 28000 cm^{-1}, then for a theoretical ΔE of 33000 cm^{-1}, you should use a laser energy of 31000 cm^{-1} to obtain the corresponding theoretical result. At the end of the ESD output, the theoretical 0-0 ΔE is printed for your information.

Fig. 5.13: The theoretical (solid black - vacuum and solid blue - water) and experimental (dashed red - water) resonant Raman spectrum for the phenoxyl radical.

OBS.: The actual Raman Intensity collected with any polarization at 90 degrees, the I($\pi/2; \|^s + \perp^s, \perp^i$) [610], can be obtained by setting RRINTES to TRUE under %ESD.

And the result is shown in Fig. 5.13. In this case, the default method VG was used. If one wants to include solvent effects, then CPCM(WATER) should be added. As can be seen, there is a noticeable difference in the main peak when calculated in water.

It is important to clarify some differences from the `ORCA_ASA` usage here. Using the ESD module, you do not need to select which modes you will account for in the spectra; we include all of them. Additionally, we can only operate at 0 K, and the maximum "Raman Order" is 2. This means we account for all fundamental transitions, first overtones, and combination bands, without including hot bands. This level of approximation is generally sufficient for most applications.

If you are working with a very large system and want to reduce calculation time, you can request RORDER 1 under the %ESD options. This setting includes only the fundamental transitions, omitting higher-order bands. This approach may be relevant, especially when including both Duschinsky rotations and the Herzberg-Teller effect, which can significantly increase computation time.

The rRaman spectra are printed with the contributions from "Raman Order" 1 and 2 separated as follows:

```
Energy                    TotalSpectrum              IntensityO1                IntensityO2
0.000000              2.722264e-08              2.722264e-08              8.
↪436299e-30
0.305176              2.824807e-08              2.824807e-08              9.
↪043525e-30
0.610352              2.931074e-08              2.931074e-08              9.
↪693968e-30
...
```

Isotopic Labeling

If you want to simulate the effect of isotopic labeling on the rRaman spectrum, there is no need to recalculate the Hessian again. Instead, you can directly modify the masses of the respective atoms in the Hessian files. This can be done by editing the $*atoms* section of the input file or directly in the Hessian file itself (see also Sec. *Isotope Shifts*). After making these adjustments, you can rerun ESD using the modified Hessian files, for example:

```
!PBE0 DEF2-SVP TIGHTSCF ESD(RR) CPCM(WATER)
%TDDFT
  NROOTS      5
  IROOT       3
END
%ESD
  GSHESSIAN  "PHE_WATER_ISO.hess"
  ESHESSIAN  "PHE_WATER_ISO.ES.hess"
END
* XYZFILE 0 2 PHE_WATER.xyz
```

As depicted in Fig. 5.14, the distinction between phenoxyl and its deuterated counterpart is evident. The peak around 1000 cm^{-1} corresponds to a C-H bond, which shifts to lower energy after deuteration. This difference of approximately 150 cm^{-1} aligns closely with experimental findings [611].

Fig. 5.14: The theoretical (solid black - C_6H_5O and solid blue - C_6D_5O) and experimental (dashed red) resonant Raman spectrum for the phenoxyl radical.

OBS: Whenever an ES Hessian is calculated using the HESSFLAG methods, it is saved in a file named BASENAME.ES.hess. If you want to repeat a calculation, you can simply use this file as an input without the need to recalculate everything.

RRaman and Linewidths

The keywords LINEW and INLINEW control the LINES function used in the calculation of the correlation function and are related to the lifetime of intermediate states and energy disordering. However, they do not determine the spectral linewidth itself, but rather the lineshape. The spectral linewidth is set independently using the RRSLINEW keyword, which defaults to 10 cm^{-1}.

It's important to note that LINEW and INLINEW significantly influence the final shape of the spectrum and should be chosen appropriately based on your specific needs. While the defaults are generally suitable, you may need to adjust them according to your requirements.

5.5.7 ESD and STEOM-CCSD or other higher level methods - the APPROXADEN option

If you plan to use the ESD module together with STEOM-CCSD, or other higher level methods such as EOM-CCSD, CASSCF/NEVPT2, some special advice must be given.

Since these methods currently do not have analytic gradients, numerical ones will be requested by default to compute the excited state geometries. This, of course, can take a significant amount of time, as they require approximately $3 \times N_{\text{atoms}}$ single-point calculations. We strongly recommend that, in these cases, you should use DFT/TD-DFT to obtain the ground/excited/triplet state geometry and Hessians, and only use the higher-level method for the final ESD step.

Also, we recommend using APPROXADEN under the %ESD options.

```
%ESD
    APPROXADEN TRUE
END
```

In this case, only one single point at the geometry of the ground state needs to be done, and the adiabatic energy difference will be automatically obtained from the ES Hessian information, without the need of a second single point at the extrapolated ES geometry, which could be unstable.

5.5.8 Circularly Polarized Spectroscopies

General Aspects of the Theory

When circularly polarized (CP) light interacts with a chiral chemical structure (optically active), it differentially absorbs left and right CP lights ($I_{LCP} \neq I_{RCP}$) resulting in the electronic circular dichroism (ECD). Similarly, it can differentially emit left and right CP lights leading to CP luminescence (CPL), which includes CP fluorescence (CPF) and phosphorescence (CPP) spectra.

Starting from ORCA 6, the ESD module has been expanded to include calculations of CP rates and spectra for chemical system. This enables the computation of the vibronic effect on ECD, CPF, and CPP spectra. The methodology is generally similar to the interaction of unpolarized light with matter, including absorption and photoluminescence. However, when a CP photon interacts with an optically active system, the electric field of the photon induces a linear displacement of the charge (transition electric dipole), while the magnetic field induces a circulation of the charge (transition magnetic dipole). These combined interactions cause an electron to be excited in a helical motion, involving both translation and rotation, along with their associated operators.

In a commonly use practice by defining a laboratory frame in which the z-axis defines the direction of the light trajectory, CP light interactions can be generated with the use of the complex vectors $\mathcal{E}\pm = \frac{1}{\sqrt{2}}(\hat{x} \pm i\hat{y})$

In this framework the FFMIO operator transforms as:

$$\mathrm{T}_{\mathrm{IF}}^{\pm} = 1/2 \sum_{j=1}^{N} \left\langle \mathrm{I} \middle| \mathrm{e}^{-\mathrm{ikr_j}} \left(\varepsilon \bullet \hat{\mathrm{p}}_{\mathrm{x}}\right) \middle| \mathrm{F} \right\rangle \pm \left\langle \mathrm{I} \middle| \mathrm{e}^{-\mathrm{ikr_j}} \left(\varepsilon \bullet \hat{\mathrm{p}}_{\mathrm{y}}\right) \middle| \mathrm{F} \right\rangle \tag{5.30}$$

In both ECD and CPL (CPF or CPP) spectroscopies the measured intensities are related to the difference of absorption or luminescence of the left and right polarized transition moments given by:

$$\Delta_{IF}^{L\pm R}(k,\epsilon) = \left|T_{IF}^{-}\right|^2 \pm \left|T_{IF}^{+}\right|^2 \tag{5.31}$$

which leads to the following expressions for the sum and the difference of the square moduli $\left|T_{IF}^{\pm}\right|^2$:

$$\Delta_{IF}^{L+R}(k,\epsilon) = 1/2 \left\langle I \left| \sum_{j=1}^{N} e^{-ikr_j} \left(\varepsilon \bullet \hat{p}_x\right) \right| F \right\rangle \left\langle I \left| \sum_{j=1}^{N} e^{-ikr_j} \left(\varepsilon \bullet \hat{p}_y\right) \right| F \right\rangle \tag{5.32}$$

$$\Delta_{IF}^{L-R}(k,\epsilon) = -\mathbf{Im} \left(\left\langle I \left| \sum_{j=1}^{N} e^{-ikr_j} \left(\varepsilon \bullet \hat{p}_x\right) \right| F \right\rangle \left\langle I \left| \sum_{j=1}^{N} e^{-ikr_j} \left(\varepsilon \bullet \hat{p}_y\right) \right| F \right\rangle \right) \tag{5.33}$$

Hence within the ED approximation, ECD and CPL radiative transition rates can be calculated through the orientational average of Equation (5.31), employing the Fermi's Golden rule:

$$k_{ECD}\left(\omega\right) = \frac{16\pi^2\omega_{ECD}}{3} \sum_F \mathbf{Im}\left(\left|\left\langle \Psi_I \left|\hat{\mu}\right| \Psi_F\right\rangle \left\langle \Psi_F \left|\hat{m}\right| \Psi_I\right\rangle\right|\right) \delta\left(E_{FI} \pm \omega_{ECD}\right) \tag{5.34}$$

$$k_{CPL}\left(\omega\right) = \frac{16{\omega_{CPL}}^3 n^2}{3\hbar c^3} \sum_F \mathbf{Im}\left(\left|\left\langle \Psi_I \left|\hat{\mu}\right| \Psi_F\right\rangle \left\langle \Psi_F \left|\hat{m}\right| \Psi_I\right\rangle\right|\right) \delta\left(E_{FI} \pm \omega_{CPL}\right) \tag{5.35}$$

where $\omega_{ECD/CPL}$ are the excitation and emission CP photon energies, respectively while ω_{FI} are the energies between the initial and final states reached in the absorption or the photoluminescent processes. Similarly E_{FI} is the transition energy and δ refers to the line-broadening mechanism arising from the lifetimes of the relevant final states and c is the speed of light. In the above expressions $\hat{\mu}$ defines electric dipole operator while $\hat{m}$ is the respective magnetic dipole operator $\hat{m} = \frac{1}{2m_e c}\sum_i r_i \times \widehat{p_i}$ and m_e is the electron mass. In the above expressions $\mathbf{Im}\left(\left|\langle\Psi_I|\hat{\mu}\right| \Psi_F\rangle \langle\Psi_F|\hat{m}\right| \Psi_I\rangle\left|\right)$ represents the rotatory strength (R_{IF}). As discussed above the transition rates including the vibronic coupling, on the Frank-Condon (FC) and Herzenberg-Teller (HT) limits, can be efficiently proceed through the path integral approach[600] in which it is possible to calculate $k_{ECD/CPL}^{obs}$ from the Fourier Transform (FT) of the respective correlation function $\chi_{(t)}$ that is computed from the path integral of the multidimensional harmonic oscillator according to[612]:

$$k_{ECD/CPL}^{obs}(\omega) = 2\alpha\mathcal{R}e\int_0^\infty \chi_{(t)} e^{\pm i\omega t} dt \tag{5.36}$$

with α being a collection of constants and for CP transition one-photon rates (ECD, CPL) considering electric dipole and magnetic dipole interactions in the expression of the rotatory strengths it takes the form:[612]

$$\chi_{(t)} = e^{\pm i\omega t}[\mathbf{Im}\left[\mu_e m_e^*\right]\rho^{FC}\left(t\right) + \sum_k \mathbf{Im}\left[\mu_e \frac{\partial m_e^*}{\partial \bar{Q}_k}\right]\rho_k^{\frac{HT}{FC}}\left(t\right) + \sum_k \mathbf{Im}\left[\frac{\partial \mu_e}{\partial \bar{Q}_k} m_e^*\right]\rho_k^{\frac{HT}{FC}}\left(t\right) + \sum_{kl} \mathbf{Im}\left[\frac{\partial \mu_e}{\partial \bar{Q}_k}\frac{\partial m_e^*}{\partial \bar{Q}_l}\right]\rho_k^{HT}\left(t\right)] \tag{5.37}$$

where μ_e and m_e represent the respective transition dipole $\langle\Psi_I|\hat{\mu}\right| \Psi_F\rangle$ and magnetic dipole $\langle\Psi_F|\hat{m}\right| \Psi_I\rangle$ moment integrals between initial and final states I, F while:

$$\rho^{FC} = Tr(e^{-i\hat{H}\tau}e^{-i\hat{H}\tau}) \tag{5.38}$$

$$\rho_k^{HT/FC} = Tr(\bar{Q}_k e^{-i\hat{H}\tau}e^{-i\hat{H}\tau}) \tag{5.39}$$

$$\rho_{kl}^{HT} = Tr(\bar{Q}_k e^{-i\hat{H}\tau}\bar{Q}_l e^{-i\hat{H}\tau}) \tag{5.40}$$

where, these traces can be evaluated following the approach discussed in Ref[600].

Finally, it is quite commonly that the ECD and CPL spectral intensities are represented against normalized absorption and photoluminescent intensities defining, similar expressions for, the dissymmetry factors g_{abs} and g_{lum}:

$$g_{abs/lum} = 2\frac{I_{LCP} - I_{RCP}}{I_{LCP} + I_{RCP}} \quad \left.\frac{4R}{D}\right|_{GS/ES\ Structure} ; \quad -2 < g_{abs/lum} < 2 \tag{5.41}$$

Implying that the associated dissymmetry spectra can also be calculated, where D and R the square of the transition dipole and the rotatory strength, respectively.

Vibration effects on ECD spectra

Vertically excited (VE) computed ECD spectra are known to often be unable to describe the experiment. This is for example the case in (R) methyl oxirane. The unshifted or shifted VE ECD BP86 computed spectra do not much the experiment in terms of shape and intensity. It has been shown that these spectra need to be computed by taking into account vibronic interactions[613].

Hence following structure optimization and frequencies calculations according to the input:

```
!BP86 DEF2-SVP TIGHTSCF OPT FREQ

* xyz 0 1
C   0.02461655377138      0.08670067686058     -5.20436273663217
C   -0.23485307714882     -0.31738971302751     -3.80610272711970
O   -0.15359212444282      1.06795113221760     -4.17749263689755
H   1.05055243293426     -0.00333310016875     -5.61325012342071
H   -0.78323750369168      0.02252140387747     -5.95969728166753
H   -1.26417590138099     -0.65347889194363     -3.56065466091728
C   0.84884023150886     -0.84537627906508     -2.89604274193698
H   0.68194744925825     -0.50794125098111     -1.85197471265376
H   0.85716491819315     -1.95559179229870     -2.89420656551675
H   1.84600617099845     -0.48435690147087     -3.21751813823758
*
```

we can compute ESD spectrum within in VE approximation and within the ESD modules according to the following input:

```
!BP86 DEF2-SVP TIGHTSCF PAL4

%TDDFT
NROOTS 10
END

%ESD
ESDFlag ECD
GSHESSIAN "c3h6o_opt_freq.hess "
PRINTLEVEL 2
DOHT TRUE
LINEW 500
SPECRANGE 40000, 70000
STATES 1,2,3,4,5,6,7,8,9,10
END

* xyzfile 0 1 c3h6o_opt_freq.xyz
```

The result is provided in Figure Fig. 5.15 where one can see that according to the expectations the computed spectrum agrees with the experiment only when FC and HT vibronic coupling schemes are taken into account

Fig. 5.15: Experimental (black) versus BP86/TDDFT (VE, blue and FC+HT red) ECD spectra for C3H6O molecule

Computation of CP-FLUOR vs CP-PHOS spectra. The case of C_3H_6O.

Following the strategy described for the computation of PL (Fluorescence (PF) of Phosphorescence (PP)) spectra in the case of C_3H_6O one can also access the respective CPL and CPP spectra.

For this one needs to compute the hessian of the 1st excited singlet (ES1) and triplet states respectively (ET0) according to the following inputs

```
!BP86 DEF2-SVP TIGHTSCF OPT FREQ

%TDDFT
NROOTS 10
```

(continues on next page)

(continued from previous page)

```
IROOT 1
IMULT 1
END

* xyzfile 0 1 c3h6o_opt_freq.xyz
```

and

```
!BP86 DEF2-SVP TIGHTSCF OPT FREQ

%TDDFT
NROOTS 10
IROOT 1
IMULT 3
SOCGRAD TRUE
TRIPLETS TRUE
END

* xyzfile 0 3 c3h6o_opt_freq.xyz
```

Then one can setup the respective PL and CPL inputs as:

PF:

```
!BP86 DEF2-SVP TIGHTSCF

%TDDFT
NROOTS 10
IROOT 1
IMULT 1
END

%ESD
ESDFlag FLUOR
GSHESSIAN "C3H6O_opt.hess"
PRINTLEVEL 2
DOHT TRUE
LINEW 500
SPECRANGE 40000, 70000
END

* xyzfile 0 1 c3h6o_opt_freq.xyz
```

CPF:

```
!BP86 DEF2-SVP TIGHTSCF

%TDDFT
NROOTS 10
IROOT 1
IMULT 1
END

%ESD
ESDFlag CPF
GSHESSIAN "C3H6O_opt.hess"
PRINTLEVEL 2
DOHT TRUE
LINEW 500
SPECRANGE 40000, 70000
END
```

(continues on next page)

(continued from previous page)

```
* xyzfile 0 1 c3h6o_opt_freq.xyz
```

PP:

```
!BP86 DEF2-SVP TIGHTSCF

%TDDFT
NROOTS 10
IROOT 1
IMULT 3
DoSOC true
TRIPLETS TRUE
SOCGRAD TRUE
END

%ESD
ESDFlag PHOSP
GSHESSIAN "C3H6O_opt_freq.hess"
TSHESSIAN "C3H6O_et0.hess"
PRINTLEVEL 2
DELE 62313
DOHT TRUE
LINEW 500
TEMP  295
SPECRANGE 40000, 70000
END

* xyzfile 0 1 C3H6O_opt.xyz
```

CPP:

```
!BP86 DEF2-SVP TIGHTSCF

%TDDFT
NROOTS 10
IROOT 1
IMULT 3
DoSOC true
TRIPLETS TRUE
SOCGRAD TRUE
END

%ESD
ESDFlag CPP
GSHESSIAN "C3H6O_opt_freq.hess"
TSHESSIAN "C3H6O_et0.hess"
PRINTLEVEL 2
DELE 62313
DOHT TRUE
LINEW 500
TEMP  295
SPECRANGE 40000, 70000
END

* xyzfile 0 1 C3H6O_opt.xyz
```

The results are summarized in Figure Fig. 5.16

Fig. 5.16: a) Computed ABS and ECD (in blue) and Florescence and CPF (in red) under FC+HT vibronic coupling schemes b) Computed Phosphorescence and CPP under FC (in blue) and FC+HT (in red) vibronic coupling schemes

Use of ABS, ECD PL and CPL as a routine analysis computational tools

Having at hand the possibility to compute the above spectroscopic properties quartet. Consisting of Absorption, ECD, Luminescent/Emission and CPL spectroscopies creates an arsenal of useful analysis computational tools. Let us consider a practical example from the
the N- bridged triarylamine heterohelicenoid chiral family of molecules, which are known to be very good CPL emitters in the CPL community. [614] Namely the R-, L- isomers of oxygen-bridged diphenylnaphthylamine for which both ABS ECD, PL and CPL experimental spectra are available [615]

In a first step one needs to compute to calculate the ECD and CPL spectra, this implies that one needs to optimize the ground state (GS) geometry and at least the GS hessian of both isomers, (see examples in *Fluorescence Rates and Spectrum* and *Vibration effects on ECD spectra*). Lets suppose that we have generated the GSHessian file R_OptFreq.hess file for the R-isomer Then one can employ the ESD to calculate the Absorption, ECD, Fluorescence and CPL (CPluorescecne) as follows.

For Absorption or ECD spectra a representative input is given by:

```
! PBE0 def2-TZVP def2/J def2-TZVP VeryTightSCF PAL8
%MaxCore 5120

%TDDFT NROOTS 5
       End

%ESD   ESDFlag     ABS or ECD
       GSHessian  "R_diphenylnaphthylamine_OptFreq.hess"
       DoHT        True
       Lines       Gauss
       InLineW     500
       STATES      1,2,3,4,5
       End

*xyz 0 1
C           0.51103659781880     -1.54799809918165     -0.46957711710367
C          -0.67682083480241     -2.19898547218227     -0.76456508506476
```

(continues on next page)

(continued from previous page)

```
C           1.59698089595014     -2.23913762447750      0.04536520831045
C           1.49004594789848     -3.60107534512753      0.24696861333386
C          -0.75027591609003     -3.56343872947031     -0.57021086417552
C           0.33765559244610     -4.27346122451739     -0.10341675327345
N           0.21984508391204     -5.66346290617628      0.07289766826829
O           2.52716581975956     -4.28847037580032      0.82258958688483
C           2.54101738078760     -5.65282427633174      0.65190453446035
C           1.43184892145219     -6.36105379222857      0.25831057061542
C           3.77147940706722     -6.27965254373750      0.89001683729049
C           3.89719111077242     -7.62085864051752      0.70851807608572
C           1.58029288205151     -7.74060974139699     -0.06585842329724
C           2.81811727341129     -8.37881977445777      0.20527726022176
O          -1.91500963727701     -4.25152600059560     -0.80702339426568
C          -2.13250715287481     -5.31470429649343      0.04909464399864
C          -1.05242665026610     -6.06056594251354      0.52598562255970
C          -3.42444532048472     -5.61905191519688      0.41037068048411
C          -3.66399173496155     -6.68754268865431      1.26210310826770
C          -1.29901325564529     -7.09753540866930      1.41085143603056
C          -2.60176522746573     -7.41491597190339      1.76502960065907
H          -4.22973438581776     -5.01558835826860      0.01489276124005
H          -4.67873778203345     -6.93777656190575      1.53843511994436
H          -0.47290745868829     -7.66765085401275      1.80929822862278
H          -2.77926863803574     -8.23774715258380      2.44386581691750
H           0.58624233693590     -0.48095844424254     -0.62562692246504
H           2.51652973024519     -1.73662559987765      0.30924801794967
H          -1.54272285055189     -1.66670869597130     -1.13102495882678
H           4.59563142014220     -5.66684912938550      1.22771972849030
H           4.83828110533620     -8.11484947410606      0.91089239524435
C           0.56233992156145     -8.49575213625774     -0.69451663429121
C           2.96252715039542     -9.75415434575818     -0.08347675768224
C           0.74593486971180     -9.81550692161063     -0.98488391710801
C           1.95147468552282    -10.46257812809982     -0.65848843248314
H          -0.36538542100730     -8.01665224889233     -0.96504181763392
H          -0.04242675451776    -10.36879326589460     -1.47770458618347
H           2.07785937833070    -11.51320955617077     -0.88255640316097
H           3.90687057621082    -10.23141194413179      0.14665882143507
*
```

For Fluorescence or CP Fluorescence spectra a representative input is given by:

```
! PBE0 def2-TZVP def2/J def2-TZVP VeryTightSCF PAL8
%MaxCore 5120

%TDDFT NROOTS 5
       End

%ESD   ESDFlag     FLUOR or CPF
       GSHessian  "R_diphenylnaphthylamine_OptFreq.hess"
       DoHT        True
       Lines       Gauss
       InLineW     500
       STATES      1,2,3,4,5
       End

*xyz 0 1
C           0.51103659781880     -1.54799809918165     -0.46957711710367
C          -0.67682083480241     -2.19898547218227     -0.76456508506476
C           1.59698089595014     -2.23913762447750      0.04536520831045
C           1.49004594789848     -3.60107534512753      0.24696861333386
C          -0.75027591609003     -3.56343872947031     -0.57021086417552
C           0.33765559244610     -4.27346122451739     -0.10341675327345
```

(continues on next page)

(continued from previous page)

```
N           0.21984508391204     -5.66346290617628      0.07289766826829
O           2.52716581975956     -4.28847037580032      0.82258958688483
C           2.54101738078760     -5.65282427633174      0.65190453446035
C           1.43184892145219     -6.36105379222857      0.25831057061542
C           3.77147940706722     -6.27965254373750      0.89001683729049
C           3.89719111077242     -7.62085864051752      0.70851807608572
C           1.58029288205151     -7.74060974139699     -0.06585842329724
C           2.81811727341129     -8.37881977445777      0.20527726022176
O          -1.91500963727701     -4.25152600059560     -0.80702339426568
C          -2.13250715287481     -5.31470429649343      0.04909464399864
C          -1.05242665026610     -6.06056594251354      0.52598562255970
C          -3.42444532048472     -5.61905191519688      0.41037068048411
C          -3.66399173496155     -6.68754268865431      1.26210310826770
C          -1.29901325564529     -7.09753540866930      1.41085143603056
C          -2.60176522746573     -7.41491597190339      1.76502960065907
H          -4.22973438581776     -5.01558835826860      0.01489276124005
H          -4.67873778203345     -6.93777656190575      1.53843511994436
H          -0.47290745868829     -7.66765085401275      1.80929822862278
H          -2.77926863803574     -8.23774715258380      2.44386581691750
H           0.58624233693590     -0.48095844424254     -0.62562692246504
H           2.51652973024519     -1.73662559987765      0.30924801794967
H          -1.54272285055189     -1.66670869597130     -1.13102495882678
H           4.59563142014220     -5.66684912938550      1.22771972849030
H           4.83828110533620     -8.11484947410606      0.91089239524435
C           0.56233992156145     -8.49575213625774     -0.69451663429121
C           2.96252715039542     -9.75415434575818     -0.08347675768224
C           0.74593486971180     -9.81550692161063     -0.98488391710801
C           1.95147468552282    -10.46257812809982     -0.65848843248314
H          -0.36538542100730     -8.01665224889233     -0.96504181763392
H          -0.04242675451776    -10.36879326589460     -1.47770458618347
H           2.07785937833070    -11.51320955617077     -0.88255640316097
H           3.90687057621082    -10.23141194413179      0.14665882143507
*
```

The results are summarized in Figure Fig. 5.17

Fig. 5.17: Black Experimental vs Calculated ABS, ECD, Fluorescence and CPF spectra for R- (in blue) and L- (in red) isomers of diphenylnaphthylamine under FC+HT vibronic coupling schemes of the $\pi \rightarrow \pi^*$ transition located at 25000 cm^{-1}.

5.5.9 Magnetic Circular Dichroism

General Aspects of the Theory

The formulation presented for the calculation of the absorption spectrum may be extended to the absorption of circularly polarized light (CPL) of a system under the effect of an external magnetic field in order to compute the MCD spectrum.[616] By assuming an electric dipolar approximation under a length formulation for the light-matter interaction, the molar absorptivity contribution by the transition between an initial state i and a final state f of one fixed oriented molecule may be expressed as:

$$\epsilon(\omega)_{if} = \alpha\omega|\mathcal{E} \cdot \langle\Psi_i|\hat{\mu}|\Psi_f\rangle|^2\delta(E_i - E_f \pm \hbar\omega) \tag{5.42}$$

where $\mathcal{E}$ is the polarization vector of the incident light, and α is a positive collection of constants.

By considering the case in which the light is propagating in the laboratory fixed $\hat{z}''$ direction, the circularly polarized light is described by $\mathcal{E} = \frac{1}{\sqrt{2}}(\hat{x}'' \pm i\hat{y}'')$ where the "$-$" sign corresponds to the left circularly polarized light and the "$+$" to the right circularly polarized light.

Similarly, as the MCD calculations presented in section *Simulation of (Magnetic) Circular Dichroism and Absorption Spectra*, the total absorptivity may be obtained by summing over all possible transitions and averaging the molecular orientations by using 3 rotation angles θ, ϕ, and χ.

$$\epsilon(\omega) = \alpha\omega \int_0^{2\pi}\int_0^{2\pi}\int_0^{\pi}\sum_{if}\frac{e^{-\frac{\epsilon_i}{k_BT}}}{Z}|\mathcal{E} \cdot \langle\Psi_i|\hat{\mu}|\Psi_f\rangle|^2 sin(\theta)\delta(E_i - E_f \pm \hbar\omega)d\theta d\phi d\chi \tag{5.43}$$

Considering χ the angle which rotates the molecule in the plane perpendicular to the magnetic field perturbation (which is colinear with the incident light propagation direction), the integrals of eq. (5.43) may be computed conveniently in a seminumerical scheme by using an intermediate r' frame.

$$\epsilon(\omega) = \alpha\omega\sum_p w_p \sum_{if}\frac{e^{-\frac{\epsilon_i}{k_BT}}}{Z}\sum_{\beta\gamma=x',y'}\overline{\overline{\mathcal{E}}}_{\beta\gamma}\langle\Psi_i|\hat{\mu}_\beta|\Psi_f\rangle\langle\Psi_f|\hat{\mu}_\gamma|\Psi_i\rangle\delta(E_i - E_f \pm \hbar\omega) \tag{5.44}$$

where θ and ϕ values are defined by a point p in a numerical grid, x', and y' are determined by θ and ϕ values according to eq. (5.45), and χ has been integrated analytically in the $\overline{\overline{\mathcal{E}}}_{\beta\gamma}$ values.

$$\begin{aligned}\hat{\mu}_{x'} &= cos(\theta)cos(\phi)\mu_{x''} + cos(\theta)sin(\phi)\mu_{y''} - sin(\theta)\mu_{z''} \\ \hat{\mu}_{y'} &= -sin(\phi)\mu_{x''} + cos(\phi)\mu_{y''}\end{aligned} \tag{5.45}$$

$$\overline{\overline{\mathcal{E}}}^{\pm}_{\beta\gamma} = \int_0^{2\pi} \mathcal{E}^{\pm}_{\beta}\mathcal{E}^{\pm *}_{\gamma} d\chi \tag{5.46}$$

Finally, by applying the Born-Oppenheimer approximation and writing the Dirac delta function in the time domain, we get:

$$\epsilon(\omega) = \alpha\omega \sum_p w_p \sum_{if} \frac{e^{-\frac{\epsilon_i}{k_B T}}}{Z} \sum_{\beta\gamma=x',y'} \overline{\overline{\mathcal{E}}}_{\beta\gamma} \langle\Theta_i|\hat{\mu}^e_\beta|\Theta_f\rangle\langle\Theta_f|\hat{\mu}^e_\gamma|\Theta_i\rangle \int e^{i(E_i-E_f-\omega)t} dt \tag{5.47}$$

and by taking the difference of absorbance between left and right CPL produce we reach an expression of the MCD intensities:

$$\Delta\epsilon(\omega) = \epsilon^-(\omega) - \epsilon^+(\omega) = -2\frac{\alpha}{Z}\omega \sum_{if}\sum_p w_p Im\Big[\int_{-\infty}^{\infty} \tilde{\chi}(t,x',y') e^{-i\hbar\omega t} dt\Big] \tag{5.48}$$

where $\tilde{\chi}$ under a first-order approximation of the transitions moments with respect to the nuclear displacement is:

$$\tilde{\chi}(t,\beta,\gamma) = e^{i\Delta E t}\Big[\hat{\mu}_{0\beta}\hat{\mu}^*_{0\gamma}\rho^{FC}(t) + \sum_k \frac{\partial\hat{\mu}_\beta}{\partial Q_k}\hat{\mu}^*_{0\gamma}\rho^{FC/HT}(t) + \sum_k \hat{\mu}_{0\beta}\frac{\partial\hat{\mu}^*_\gamma}{\partial Q_k}\rho^{FC/HT}(t) + \sum_{kl}\frac{\partial\hat{\mu}_\beta}{\partial Q_k}\frac{\partial\hat{\mu}^*_\gamma}{\partial Q_l}\rho^{HT}(t)\Big] \tag{5.49}$$

Similarly, as section *Simulation of (Magnetic) Circular Dichroism and Absorption Spectra*, the transition moments under the effect of an external magnetic field perturbation may be estimated by using a QDPT (eq. (5.184)), and the derivatives are approximated numerically in a similar way as the ESD-Absorption case.

By applying a quasi-degenerate perturbative theory, similar to the inclusion of spin-orbit coupling effects in the phosphorescence calculations, the effect of an external magnetic field may be included in the representation of the quantum states [616]. As a result, the differential absorption of left and right circularly polarized light may be computed to obtain the vibrationally-corrected magnetic circularly dichroism spectrum. The input for the calculation is similar to the absorption case described above; nevertheless, ESD(MCD) should be used. Additionally, the intensity of the external magnetic field "B" (in Gauss) should be included, and a Lebedev grid for a semi-numerical molecular orientational average should be selected.

The method is only available with an electronic structure generated by TDDFT. The calculation supports the inclusion of Herzberg-Teller effects by setting DOHT TRUE, the ground state Hessian needs to be provided similarly to the absorption case, while the excited state Hessian can be provided or computed under a no external magnetic field approximation. An input example is:

```
!B3LYP DEF2-TZVP TIGHTSCF ESD(MCD)

%TDDFT NROOTS 40
       TDA FALSE
END

%ESD    GSHESSIAN "pbq.hess"
        Hessflag AHAS
        DOHT    TRUE
        STATES 1
        B 50000.0
END

* xyzfile 0 1 pbq.xyz
```

Similarly, to the ESB(ABS) calculation the MCD spectrum is saved in a BASENAME.MCD file as:

```
Energy          TotalSpectrum           IntensityFC             IntensityHT
4817.11         -6.324671e-05           -4.528264e-06           -5.871844e-05
5026.55         -6.717718e-05           -4.809014e-06           -6.236816e-05
5235.99         -7.126386e-05           -5.100843e-06           -6.616302e-05
5445.43         -7.551756e-05           -5.404513e-06           -7.011304e-05
5654.87         -7.994996e-05           -5.720850e-06           -7.422911e-05
5864.31         -8.457379e-05           -6.050750e-06           -7.852304e-05
...
```

Specific Keywords and recommendations

Once selected the ESD(MCD) calculation two variables need to be defined in the `%esd` block:

- The intensity of the magnetic field in Gauss "B"
- The grid to make the molecular orientational average "LEBEDEVINTEGRATIONPOINTS"

The MCD signals use to be much more sensitive than the corresponding absorption intensities. As result, the MCD calculations are much more sensitive to errors in the electronic structure. So it is highly recommended to first fully understand the electronic structure problem and verify there are no important problems. In this direction we suggest the following recommendations:

- Be sure the obtained electronic states are in the correct order by assigning point group symmetry labels and comparing them with better electronic structure methods. Due to the MCD intensities emerging as a result of the interaction of states by the magnetic field perturbation, a wrong-located state in energy may affect the MCD intensities, even if it is not in the energy range you are computing. We do not recommend using the method as a black box.
- It is recommended to do first an ESD-absorption calculation on the exact same level of theory, verify the intensities and also solve any problem related to the PES representation.
- The lineshape for the best agreement of the MCD intensities compared against the experimental measurements may differ for the best value for absorption intensities.

5.5.10 Tips, Tricks and Troubleshooting

- Currently, the ESD module works optimally with TD-DFT (Sec. *Excited States Calculations*), but also with ROCIS (Sec. *Excited States with Restricted Open-shell CIS - ROCIS*), EOM/STEOM (Sec. *Excited States via EOM-CCSD* and Sec. *Excited States via STEOM-CCSD*) and CASSCF/NEVPT2 (Sec. *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)* and Sec *N-Electron Valence State Perturbation Theory (NEVPT2)*). Of course you can use any two Hessian files and input a custom DELE and TDIP obtained from any method (see Sec. *Excited State Dynamics*), if your interested only in the FC part.
- If you request for the HT effect, calculating absorption or emission, you might encounter phase changes during the displacements during the numerical derivatives of the transition dipole moment. There is a phase correction for TD-DFT and CASSCF, but not for the other methods. Please be aware that phase changes might lead to errors.
- Please check the K*K value if you have trouble. When it is too large (in general larger than 7), a warning message is printed and it means that the geometries might be too displaced and the harmonic approximation might fail. You can try removing some modes using TCUTFREQ or use a different method for the ES PES.
- If using DFT, the choice of functional can make a big difference on the excited state geometry, even if it is small on the ground state. Hybrid functionals are much better choices than pure ones.
- In CASSCF/NEVPT2, the IROOT flag has a different meaning from all other modules. In this case, the ground state is the IROOT 1, the first excited state is IROOT 2 and so on. If your are using a state-averaged calculation with more than one multiplicity, you need also to set an IMULT to define the right block, IMULT 1 being the first block, IMULT 2 the second and etc.
- If using NEVPT2 the IROOT should be related to the respective CASSCF root, don't consider the energy ordering after the perturbation.

- After choosing any of the HESSFLAG options, a BASENAME.ES.hess file is saved with the geometry and Hessian for the ES. If derivatives with respect to the GS are calculated, a BASENAME.GS.hess is also saved. Use those to avoid recalculating everything over and over. If you just want to get an ES PES, you can set WRITEHESS TRUE under
- Although in principle more complete, the AH is not NECESSARILY better, for we rely on the harmonic approximation and large displacements between geometries might lead to errors. In some cases the VG, AHAS and so one might be better options.
- If you use these .hess files with derivatives over normal modes in one coordinate system, DO NOT MIX IT with a different set of coordinates later! They will not be converted.
- Sometimes, low frequencies have displacements that are just too large, or the experimental modes are too anharmonic and you might want to remove them. It is possible to do that setting the TCUTFREQ flag (in cm^{-1}), and all frequencies below the given threshold will be removed.
- If you want to change the parameters related to the frequency calculations, you can do that under %FREQ (Sec. *Vibrational Frequencies*). The numerical gradient settings are under %NUMGRAD (Sec. *Numerical Gradients*).
- When computing rates, the use of any LINES besides DELTA is an approximation. It is recommended to compute the rate at much smaller lineshape (such as 10 cm^{-1}) to get a better value, even if the spectrum needs a larger lineshape than that.
- When in doubt, try setting a higher PRINTLEVEL. some extra printing might help with your particular problem.

5.5.11 More on the Excited State Dynamics module

ORCA has now a module designed to calculate properties related to excited states named `ORCA_ESD`. It can be used to predict absorption/emission spectra, transition rates, resonant Raman, and MCD spectra, based on a path integral approach to the dynamic process [600]. It has some of the functionalities of `ORCA_ASA` and even more, as it will be discussed. What we do here is NOT a conventional dynamics with trajectories along time points, we rather solve the equation for the transition rates or intensities depending on the different cases considered.

This formulation works because there is an analytic solution to the path integral of the Multidimensional Harmonic Oscillator and the assumption of Harmonic nuclear movement is critical. In many cases that approximation does hold and the results are in very good agreement with the experiment. The general usage of the `ORCA_ESD` module and some examples are already presented on Sec. *Excited State Dynamics* and it is recommended to read that before going into the details here. We now will discuss the specifics and keywords related to of each part of the module. A complete keyword list can be found at the end of this section.

Absorption and Emission Rates and Spectrum

General Aspects of the Theory

The idea behind calculating the absorption or emission rates starts with the equation from the quantization of the electromagnetic field for the transition rates between and initial and a final state:

$$k(\omega)_{if} = \frac{4\omega^3 n^2}{3\hbar c^3} |\langle \Psi_i | \widehat{\mu} | \Psi_f \rangle|^2 \delta(E_i - E_f \pm \hbar\omega) \tag{5.50}$$

with $\hbar\omega$ being the energy of the photon, $\widehat{\mu}$ the dipole operator and n the refractive index of the solvent, as suggested by Strickler and Berg [604].

One way to obtain $k(\omega)$ is to compute it in the frequency domain, by calculating the Franck-Condon Factors between all initial and final states that satisfy the Dirac delta in Eq. *(5.51)*, considering the thermally accessible initial states with the appropriate weight,

$$k^{obs} = \int k(\omega) d\omega, \qquad k(\omega) = \sum_{if} P_i(T) k_{if}(\omega), \tag{5.51}$$

where $P_i(T) = e^{-\frac{\epsilon_i}{k_B T}}/Z$ is the Boltzmann population of a given initial state at temperature T, ϵ_i is the total vibrational energy of state i and Z is the vibrational partition function. However, this can lead to a very large number of states to be included, particularly if there are low frequency modes. In this work we will take the a different approach and switch to the time domain, by using the Fourier Transform representation of the Dirac delta,

$$\delta(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{i\omega t}dt, \tag{5.52}$$

so that the equation to solve, in atomic units, is:

$$k(\omega) = \frac{2\omega^3}{3\pi c^3 Z}\sum_{if} e^{-\frac{\epsilon_i}{k_B T}}\langle\Theta_i|\vec{\mu}^e|\bar{\Theta}_f\rangle\langle\bar{\Theta}_f|\vec{\mu}^e|\Theta_i\rangle\int e^{i(E_i-E_f-\omega)t}dt,$$

with $\vec{\mu}^e$ being the "electronic transition dipole" and $|\Theta\rangle$ the vibrational wavefunction of the initial or final state.

After some extra steps, redefinition of the time variable and insertion of a resolution of identity, it can be shown that this equation is ultimately simplified to a Discrete Fourier Transform (DFT) of a correlation function $\chi(t)$ with a timestep Δt, multiplied by a prefactor α [600]:

$$\begin{aligned} k(\omega) &= \alpha\int Tr(\vec{\mu}^e e^{-i\hat{H}\tau}\vec{\mu^e}e^{-i\hat{\bar{H}}\bar{\tau}})e^{i\Delta E t}e^{-i\omega t}dt \\ &= \alpha\int \chi(t)e^{-i\omega t}dt \\ &= 2\alpha\ \mathfrak{Re}\int_0^{\infty}\chi(t)e^{-i\omega t}dt \\ &\simeq 2\alpha\Delta t\ \mathfrak{Re}\ DFT\{\chi(t)\}, \end{aligned}$$

and this correlation function is then calculated using path integrals analytically at each time point t.

If one considers that the electronic part of the transition dipole varies with nuclear displacements and we allow for it to depend on the normal coordinates (**Q**), such as:

$$\vec{\mu}^e(\mathbf{Q}) = \vec{\mu}_0^e + \sum_i \left.\frac{\partial\vec{\mu}^e}{\partial Q_i}\right|_{\mathbf{Q}=0} Q_i + \ldots, \tag{5.53}$$

we can even include vibronic coupling or the so-called Herzberg-Teller (HT) effect. The Frank-Condon (FC) approximation keeps only the coordinate-independent term. The correlation function for the HT cases can then be derived recursively from the FC one and the calculation is done quite efficiently. It is important to say the one must choose ONE set of coordinates in order to expand the transition dipole. In our formulation, it is always that of the FINAL state and that has some implications discussed below.

Other important aspect of this theory is that, in order to solve the path integrals, one has to work in one set of coordinates, the initial (**Q**) or the final state ones ($\bar{\mathbf{Q}}$, with the bar indicating final coordinates). As we have a transition matrix element, one set of coordinates have to be transformed into the other and it is easy to show that they are related by

$$\mathbf{Q} = \mathbf{J}\bar{\mathbf{Q}} + \mathbf{K}. \tag{5.54}$$

That was first proposed by Duschinsky in the late 1930s[603] with the Duschinsky rotation matrix **J** and the displacement vector **K** defined as

$$\mathbf{J} = \mathbf{L}_x^{\mathrm{T}}\bar{\mathbf{L}}_x, \qquad \mathbf{K} = \mathbf{L}_x^{\mathrm{T}}(\bar{\mathbf{q}}_0 - \mathbf{q}_0),$$

with $\mathbf{L}_x$ being the matrix containing the normal modes, here described in Cartesian coordinates (x), and $\mathbf{q_0}$ begin mass weighted coordinates ($q_i = \sqrt{m_i}x_i$).

The program runs by first reading and obtaining the initial and final state geometries and Hessians, then computes the Duschinsky rotation matrix and displacement vector, calculates the derivatives for the transition dipoles and computes the correlation function. After that, the DFT is done and the rates are obtained and printed when necessary. As the intensities observed experimentally are proportional to the rates, the spectrum is also calculated and printed on a BASENAME.spectrum file. If PRINTLEVEL HIGH is requested under %ESD, the correlation function is also printed on a BASENAME.corrfunc file.

OBS: The units for the Emission spectra are rather arbitrary, but for Absorption they are the experimental "molar absorptivity (ε)" in L mol^{-1}cm^{-1} [600]. Be aware that these are dependent on the line width of the curves.

Approximations to the excited state PES

As already mentioned in Sec. *Excited State Dynamics*, in order to predict the rates we need at least a ground state (GS) and an excited state (ES) geometry and Hessian. In ORCA, we have seven different ways to approximate this ES PES: AHAS, VH, VG, HHBS, HHAS, UFBS and UFAS. Those can can be choosen by setting the HESSFLAG under %ESD. If you actually optimize the ES geometry and input the Hessian, that will be called an Adiabatic Hessian (AH) method an no keyword must be given on the input.

OBS: In the present version, these approximations are only available for Absorption, Fluorescence and resonant Raman. They cannot be directly used for phosphorescence and ISC rate calculations. However, one can do the latter calculations by generating approximate ES Hessians from "fake" fluorescence or absorption runs, and then using the ES Hessians in AH calculations (vide infra).

The idea behind these approximations is to do a geometry update step ($?\mathbf{S} = -\mathbf{g}\mathbf{H}^{-1}$ for Quasi-Newton and $?\mathbf{S} = -\mathbf{g}(\mathbf{H} + \mathbf{S})^{-1}$ for Augmented Hessian) to obtain the ES structure and somehow approximate the ES Hessian. The gradient ($\mathbf{g}$) and Hessian ($\mathbf{H}$) used on the step are on column Step of Table Table 5.5 below, with a description of the final ES Hessian:

Table 5.5: Methods used to estimate the ES PES

Method	Step	ES Hessian
AHAS	ES grad + GS Hessian	calculated on the ES geometry
VH	ES grad + ES Hessian at GS geometry	calculated on the GS geometry
VG (default)	ES grad + GS Hessian	equal to GS Hessian
VGFC	ES grad + GS Hessian	equal to GS Hessian (+ APPROXADEN TRUE)
HHBS	ES grad + Hybrid ES Hessian on GS geometry	Hybrid Hessian on GS geometry
HHAS	ES grad + GS Hessian	Hybrid Hessian on ES geometry
UFBS	ES grad + Updated frequencies ES Hessian on GS geometry	Updated frequencies ES Hessian on GS geometry
UFAS	ES grad + GS Hessian	Updated frequencies ES Hessian on ES geometry

OBS: Always use the GS geometry on the input file, equal to the one in the GSHESSIAN! If you asked for OPT FREQ at the input, a .xyz file is generated with the same geometry found on the .hess. If you picked the geometry from the .hess file, remember that it is in atomic units, so you have to use BOHRS on the main input.

After the calculation of the ES PES, a file named BASENAME.ES.hess is printed and can be used in future calculations. For example, one can use it in a separate AH calculation, as if the file contains the optimized ES geometry and the exact ES Hessian (of course, in reality both are approximate). This allows one to perform e.g. phosphorescence and ISC rate calculations with approximate ES PESs even though ESD does not support these calculations directly. For example, the phosphorescence rate from a triplet state to the S_0 state, with the VG approximation for the triplet PES and the ES gradient computed at the TDDFT level, can be computed as follows:

- Run an ESD fluorescence calculation using the VG approximation, but where "IROOTMULT TRIPLET" is added to the %TDDFT block. The resulting rate and spectrum are not meaningful, but the BASENAME.ES.hess file generated from this calculation is the correct VG approximation to the triplet PES.
- Run an ESD phosphorescence calculation using the AH method, with the aforementioned BASENAME.ES.hess file as the TSHESSIAN.

This way, the first ESD calculation only uses the information of the scalar triplet state in computing the VG step, while the second ESD calculation only uses the SOC-corrected states. The same ES Hessian file can therefore be used in the calculations of all three spin sublevels of the triplet state.

In addition to the BASENAME.ES.hess file, if there was any updates on the GS Hessian, like transition dipole derivatives, a BASENAME.GS.hess is also printed.

- The step can be controlled with the GEOMSTEP flag, with QN or AUGHESS options.

- Currently all ES Hessians are calculated numerically, if you want to change the parameters related to the frequency calculations, you can do that under %FREQ (Sec. *Vibrational Frequencies*). The numerical gradient settings are under %NUMGRAD (Sec. *Numerical Gradients*).
- The ES hybrid Hessian is calculated in the same way as described in Sec. *Hybrid Hessian*, except that the "model" Hessian is the GS one.
- The ES Hessian with updated frequencies is recalculated from the same GS normal modes, after an update on the frequencies, as $\mathbf{H}_{up} = \mathbf{L}\omega_{up}^2\mathbf{L}^T$. With $\mathbf{L}$ being the normal modes and ω_{up} the updated frequencies, with negative sign being kept after the square.
- The frequencies are updated depending on a calculation of the energy after a given step. If the ES modes are equal to the GS, then a step over a coordinate δq_i that would result in an energy difference δE is given by $\delta q_i = (-\frac{g_i}{\sqrt{\omega_i}} + \frac{\sqrt{g_i^2}}{\omega_i} + 2\delta E\omega_i)/\omega_i$. The default δE used is 10^{-4} Eh, in general above the error of the methods. If the error in energy after the step is larger than a threshold given by the UPDATEFREQERR flag (default 0.20 or 20%), the gradients are calculated and the frequency recomputed. If not not, that mode and frequency are assumed to be the same.
- The Updated Frequencies method can greatly accelerate the calculation of the Hessian, for much fewer gradient calculation are necessary, although you do not correct the modes. Also, the derivatives over the modes are already computed simultaneously.
- The expected energy error δE can be changed using the UF_DELE flag.
- The default method is the VG, but the AHAS is more trustworthy for unknown systems, although a lot heavier (Sec. *A better model, Adiabatic Hessian After a Step (AHAS)* and [600]).
- Always check the sum of K_i^2 printed on the output. If that number is too high (above 8 or so), it means that the geometries are too displaced and the theory might not work on these cases (check for different coordinate systems then, Sec. *Normal modes coordinate systems*).
- Also check for RMSD between the geometries after a step. If the difference is too big, there might be problems with the step.

Mixing methods

In principle, it is possible to use different methods to compute different parts needed for the `ORCA_ESD` module. You could, for instance use (TD)DFT analytical gradients for the ground/excited state geometries and Hessians and a more elaborate method such as STEOM-CCSD to get the energies and transition dipoles. If you want to do that, just input the Hessians and use the DELE flag for the energy difference between the states - at their own geometry! - and TDIP x,y,z to input the transition dipole. If there is SOC and the transition dipole is complex, use TDIP x.real, x.imag, y.real, y.imag, z.real, z.imag. The program will automatically detect each case. If you don't input these, they will be obtained by the module during the run, so you can set the excited method you want and let the program figure out DELE and TDIP.

OBS: It is not advisable to mix different levels of theory during a geometry step though. If you obtained a GS Hessian from DFT, doing a step based on a CASSCF ES numerical gradient might not lead to reasonable results. The same would be problematic even for different DFT functionals.

Removal of frequencies

If, after calculating an ES Hessian you end up with negative frequencies, the calculation of the correlation function might run into trouble. The default for the module is to turn all negative frequencies positive, printing a warning if any of them was lower than -300 cm^{-1}. If that is the case, you are probably on a saddle point and not even a minimum, so results should be taken with care,

You can also choose to completely remove the negative frequencies (and the corresponding from the GS), by setting IFREQFLAG REMOVE or leave them as negative with IFREQFLAG LEAVE under %ESD.

Sometimes, low frequencies have displacements that are just too large (check on the **K** vector), or the experimental low frequency modes are too anharmonic and you might want to remove them. It is also possible to do that setting the TCUTFREQ flag (in cm^{-1}), and all frequencies below the given threshold will be removed.

Normal modes coordinate systems

When working with systems with weak bonds, such as hydrogen bonds and π stacking, or with biphenyls and similar planar molecules it is common that there will be low frequency-high amplitude modes related to the angular bending. It has been shown that, in some cases, the normal modes transformed from Cartesian coordinates might be not sufficient to describe systems with these large amplitude motion [617]. In those, the definition of normal modes in terms of some curvilinear set of coordinates such as the internal ones are more suitable.

The first order transformation from Cartesian ($\mathbf{x}$) to internal ($\mathbf{s}$) coordinates is given by Wilson's **B** matrix[618] as

$$\mathbf{s} = \mathbf{B}(\mathbf{x} - \mathbf{x_0}), \tag{5.55}$$

and here we use Baker's[619] delocalized internal coordinates as default. First, a redudant set is build and an singular value decomposition of the $\mathbf{G} = \mathbf{B}\mathbf{B}^T$ matrix is performed to obtain the non-redundant set. The latter can be generated by $\mathbf{B}' = \mathbf{U}^T\mathbf{B}$, where **U** are the eigenvectors correponding to non-zero eigenvalues of **G**. Then an orthogonal set is contructed from $\mathbf{B}'' = \mathbf{G}'^{-1/2}\mathbf{U}^T\mathbf{B}$. As the eigenvectors are not conitnuous functions of the coordinates, in order to avoid using an arbitrary selection, we will follow the work of Reimers[620] and set $\mathbf{G}^{-1/2}\mathbf{U}^T = \tilde{\mathbf{G}}^{-1/2}\tilde{\mathbf{U}}^T$, or use the same transformation for the initial an final coordinates. Please note that this may lead to numbers larger than 1 on the Duschinsky rotation matrix, for it is an approximation and the calculated rates may vary a little. The normal modes in internal coordinates ($\mathbf{L}_s$) are then obtained from those in Cartesian ones ($\mathbf{L}_x$) as

$$\mathbf{L}_s = \mathbf{B}''\mathbf{M}^{1/2}\mathbf{L}_x, \tag{5.56}$$

and the Duschinsky relation ((5.54)) still holds[617], with the displacement vector being simply

$$\mathbf{K}_s = \mathbf{L}_s^T(\tilde{\mathbf{s}} - \mathbf{s}). \tag{5.57}$$

The options available for coordinate systems can be set under COORDSYS, and can be CARTESIAN, INTERNAL (for Baker delocalized - default), WINT (for weighted internals following Swart and Bickelhaupt [621]), FCWL (force constant weighted following Lindh [622]) and FCWS (same as before, but using Swart's model Hessian).

OBS: Calculating in internal coordinates is usually better but not necessarily. If something goes wrong, you may also want to try the Cartesian option.

Geometry rotation and Duschinsky matrices

The electronic transition should not to take place simultaneously with translations and rotations[623] of the molecular structure. Before further calculations take place, the initial and final state structures are superimposed to satisfy Eckart's conditions by obtaining a rotation matrix $\mathbb{B}$ that ensures $\sum m_i(\mathbb{B}\mathbf{R_i} \times \tilde{\mathbf{R}}_\mathbf{i}) = 0$ [624], with **R** being Cartesian coordinates. As the initial geometry is rotated, so must be the corresponding normal modes $\mathbf{L}_x$ also. This can be turned of by setting the flag USEB FALSE.

By the default the Duschinsky rotation matrix is set to Identity, to take advantage of our much faster algorithm. To turn that on, just set USEJ TRUE. You can check the "diagonality" of this matrix by looking at the Diagonality Index (DI), here defined as $\sqrt{\sum_i \mathbf{J}_{ii}^2 / \sum_{ij} \mathbf{J}_{ij}^2}$. A DI=1 would be a perfectly diagonal matrix. The amount of mixing between modes is rpresented by the Mixing Index, with is here is given by $\langle |J_{max}| \rangle$, or the average value of the maximum J_i of each line. If DI=1, it means each normal coordinate from the initial state is equal to a mode of the final state. When USEJ=TRUE, the largest components of the **J** matrix are printed along with the **K** vector, so you can have a better idea of how the mixing is occuring.

Derivatives of the transition dipole

The derivatives of the transition dipoles with respect to the normal coordinates of the final state can be obtained directly from the derivatives with respect to the Cartesian coordinates as

$$\mathbf{U}(\bar{\mathbf{Q}}) = \bar{\mathbf{L}}_x^T \mathbf{M}^{-1/2} \mathbf{U}(\bar{\mathbf{R}}), \tag{5.58}$$

$\mathbf{U}$ being the matrix of the x,y and z components of the derivative, $\mathbf{M}$ a $3N \times 3N$ matrix with the atomic masses along the diagonal. Also, in case one already has the derivatives with respect to the initial state , those can be transformed into the derivatives with respect to the final state by using the Duschinsky relation, assuming that $\vec{\mu}_0^e(\bar{\mathbf{Q}}) + \sum_i \frac{\partial \vec{\mu}^e}{\partial \bar{Q}_i} \bar{Q}_i = \vec{\mu}_0^e(\mathbf{Q}) + \sum_i \frac{\partial \vec{\mu}^e}{\partial Q_i} Q_i$, so that

$$\frac{\partial \vec{\mu}^e}{\partial \bar{Q}_k} = \sum_j \mathbf{J}_{jk} \frac{\partial \vec{\mu}^e}{\partial Q_j}. \tag{5.59}$$

By default, this transformation is NOT done, since Eq. (5.59) is an approximation. If you want to turn it on, set CONVDER TRUE under %ESD.

OBS: Remember that, if you already have the Cartesian derivatives over the final state, like if you use AHAS for an absorption spectrum, the conversion should be exact (although there might be numerical issues, always use a larger GRAD for frequencies!).

Alternatively, these can be calculated numerically from displacements over each normal mode. In this case, it is convenient to use the dimensionless normal coordinates $q_i = \omega_i^{1/2} Q_i$ which represent proportional displacements on the PES [625]. We use $\Delta q = 0.01$ by default, but this can be changed setting the DER_DELQ flag.

- Again, DO NOT MIX different coordinates systems. If the derivatives were calculated over one coordinate set and you decide to change it, it has to be recalculated. You can manually delete them from the BASENAME.ES.hess file.
- For hybrid functionals, you can choose to use DFT for the gradient, energy and transition dipole, and the fast simplified TDA (Sec. *Simplified TDA and TD-DFT*) only for the derivatives by seting STDA TRUE under %ESD.
- A simple trick can be used to accelerate the computation of derivatives. If the first displacement gives a transition dipole that is too close to the reference, then the derivative can be assumed to be small and just the plus displacement may be taken to compute the derivative (with an usually small error). If it is large enough, then the minus displacement is also done and central differences is used. This is the default method and can be turned off by setting FASTDER to FALSE.
- The central differences option can be altogether turned off by setting CENTRALDIFF FALSE under %ESD.
- If you are having problems, set a larger PRINTLEVEL to check the individual calculation of the derivatives, you might be having some kind of root flipping during the displacement, or some other issue.

The Fourier Transform step

After the calculation of the correlation function, it is necessary to do a Fourier Transform (FT) step. To do that numerically, it is needed to correctly choose the grid in wich the time points will be computed, for that affects how the results will be obtained in the frequency domain. We have developed a method to generate an optimal set of parameters, depending on the final spectral resolution desired [600] and it will be used by default. Even so, you can choose your own grid by setting the NPOINTS and MAXTIME (in atomic units!) flags under %ESD. There are a few comments related to that:

- Because we solve the FT using a very efficient Cooley-Tukey algorithm, the NPOINTS should be always multiple of two. You can put any number on the input, but the next larger multiple of two will be calculated and set.
- The MAXTIME should be enough so that the correlation function goes to zero. If anything goes wrong, please check the BASENAME.corrfunc file for that.
- The finer the spectral resolution chosen with SPECRES, the largest MAXTIME must be.

- If you have a larger MAXTIME, you also must increase NPOINTS, otherwise the grid will be too sparse and many oscillations will be skipped.

Spectrum options

The final spectrum is saved in a BASENAME.spectrum file, with the total spectrum, the FC and HT parts discriminated, as explained in Sec. *Excited State Dynamics*. He are some detailed about it:

- The range for which the spectrum is saved is given by default, but it can be set using SPECRANGE flag under %ESD, as SPECRANGE 10000,70000.
- All of the INPUT units should always be in CM-1, but you can choose the OUTPUT units by setting the UNIT flag to CM-1, NM or EV.
- In order to better converge the correlation function and approximate experimental spectra, a lineshape function can be used instead of the delta. The default is to use a LOREZENTIAN lineshape, but LINES can be set to DELTA, LORENTZ, GAUSS or VOIGT.
- The DELTA lineshape might lead to a correlation function that oscillates forever, so please take care with that option.
- The default line widths are LINEW 50 and INLINEW 250.
- If you use a VOIGT lineshape, the Gaussian width can be controlled separately using the INLINEW flag. By default, it will be proportial to the Lorezntian to reach the same FWHM.
- The LINEW and INLINEW are NOT the full width half maximum ($FWHM$) of these curves. However they are related to them by: $FWHM_{lorentz} = 2 \times LINEW$ and $FWHM_{gauss} = 2.355 \times INLINEW$. For the VOIGT curve, it is a little more complicated but in terms of the other FWHMs, it can be aproximated as $FWHM_{voigt} = 0.5346 \times FHWM_{lorentz} + \sqrt{(0.2166 \times FWHM^2_{lorentz} + FWHM^2_{gauss})}$.
- The resolution of the spectrum can be modified with the SPECRES flag. By default it is a fraction of the LINEW. Please be aware that higher resolution (smaller SPECRES), mean a larger grid for the correlation function and more time points to calculate on.

General

- The temperature is accounted for exactly on Absorption and Emission [600] and can be set using the TEMP flag.
- PRINTLEVEL can be set to HIGH in order to print more details, including Huang-Rhys factors which are useful for rationalizing the contribution of different vibrational modes to the rate/spectrum.
- The frequencies read from the Hessian files can be scaled by any number by setting the SCALING flag under %ESD. The default is 1.0.
- If necessary, the transition dipole can also be scaled by setting the TDIPSCALING flag.
- If you just want to compute an ES PES and stop, set WRITEHESS to TRUE and the correlation function will be skipped.
- In order to make use of the fastest algorithm, set SAMEFREQ to TRUE and the DO method will be used, assuming equal Hessians between initial and final states and maximizing the efficiency when calculating the correlation function.
- If you want to calculate phosphorescence rates, you MUST input the adiabatic energy difference DELE manually (the energy difference between each state at its own geometry). And, of course, don't forget to set the SOC module to true.

Intersystem crossing rates

General Aspects of the Theory

Intersystem crossing (ISC) rates between a given initial state i and a final state f can be calculated from Fermi's Golden rule:

$$k(\omega)_{if} = \frac{2\pi}{\hbar}|\langle\Psi_f|\widehat{H}_{SO}|\Psi_i\rangle|^2\delta(E_i - E_f), \tag{5.60}$$

which is quite similar to the Eq. (5.50) for Fluorescence, except for the frequency term. The same trick used there can be applied here to swtich to the time domain. Then, we are left with a simple time integration, which is not anymore difficult to solve than the equations above.

One can use all of the infrastructure already presented to compute these ISC rates, including Duschisnky rotation, vibronic coupling effects, use of different coordinate systems and so on. Right now, its use is optimized for CIS/TDDFT, as explained in Section *Intersystem Crossing Rates (unpublished)*, but it can be applied in general by combining simpler methods to obtain the geometries and Hessians with more advanced methods to compute the SOC matrix elements, when needed.

Tips and Tricks

- The DELE must be given when using ESD(ISC), it is not automatically computed.That is the energy of the initial state minus the energy of the final state, each at its own geometry.
- A SOC matrix element calculated from any method can be given on the input using the SOCME Re, Im flag, where these are the real and imaginary parts of that number.
- The SOCMEs from TD-DFT are not bad, maybe except for those between the ground singlet and the triplets. In that case, a multireference calculation might be the preferred option.
- If the final state is higher in energy than then initial state, the DELE is a negative number. In that case, there is barrier to go up when doing the ISC and the rates becomes more sensitive to the temperature.
- In contrast to Fluorescence, the ISC rates depend strongly on the inclusion of Duschisnky rotations, please take care when using USEJ FALSE.
- The default LINES is GAUSS, and the default INLINEW of 250 cm^{-1} might be too large. One should always investigate it by varying the width a bit. Other LINES can increase the error, please take care when changing it.

Resonant Raman Spectrum

General Aspects of the Theory

Raman intensities can be obtained in many different ways, depending on the experimental set up. As discussed extensively by D. A. Long [609, 610], the part of it that is "set up independent" is the Scattering Factor (or Raman activity), given by:

$$S = 45a^2 + 7\gamma \quad \left(\frac{C^2m^2}{amuV^2}\right), \tag{5.61}$$

where the a is related to the isotropic part of the "transition polarizability" between an initial state and a final state with a different vibrational quantum number $\langle\Psi_f|\hat{\alpha}|\Psi_i\rangle = \alpha_{if}$:

$$a = \frac{1}{3}|(\alpha_{xx} + \alpha_{yy} + \alpha_{zz})|$$

and γ is also related to its off-diagonal elements:

$$\gamma = \frac{1}{2}[|(\alpha_{xx} - \alpha_{yy})|^2 + |(\alpha_{yy} - \alpha_{zz})|^2 + |(\alpha_{zz} - \alpha_{xx})|^2 + 6(|\alpha_{xy}|^2 + |\alpha_{yz}|^2 + |\alpha_{xz}|^2)].$$

This transition polarizability itself can be computed using Kramers, Heisenberg and Dirac (KHD) formalism and it can be shown that each of its Cartesian components can be calculated as a sum-over-states:

$$\alpha_{\rho\sigma}^{if} = \frac{1}{\hbar} \sum_{n \neq i,f} \left(\frac{\langle \Psi_f | \mu_\rho | \Psi_n \rangle \langle \Psi_n | \mu_\sigma | \Psi_i \rangle}{\Delta E_{ni} - \omega_{laser} + i\gamma_{lt}} + \frac{\langle \Psi_f | \mu_\rho | \Psi_n \rangle \langle \Psi_n | \mu_\sigma | \Psi_i \rangle}{\Delta E_{ni} + \omega_{laser} + i\gamma_{lt}} \right) \tag{5.62}$$

In (5.62), the sum is over any number of electronic excited states n, ΔE_{in} is the energy difference between the initial state and the excited, ω_{laser} is the laser energy and γ_{lt} is the lifetime broadening to avoid a zero on the denominator. If we work with a laser for which the frequency is close to the excited state energy difference, the first term is much larger than the second and can approximate alpha by

$$\alpha_{\rho\sigma}^{if} \simeq \frac{1}{\hbar} \sum_{n \neq i,f} \left(\frac{\langle \Psi_f | \mu_\rho | \Psi_n \rangle \langle \Psi_n | \mu_\sigma | \Psi_i \rangle}{\Delta E_{ni} - \omega_{laser} + i\gamma_{lt}} \right). \tag{5.63}$$

This equation can be solved using a path integral approach by switching to the integral form of $1/x$:

$$\frac{1}{x} = \frac{i}{\hbar} \int_0^\infty e^{-ixt/\hbar} dt \tag{5.64}$$

So that the components of α_{if} can be given by:

$$\alpha_{\rho\sigma}^{if} \simeq \sum_{n \neq i,f} \frac{i}{\hbar^2} \int_0^\infty \langle \Psi_f | \mu_\rho | \Psi_n \rangle \langle \Psi_n | \mu_\sigma | \Psi_i \rangle e^{-it(\Delta E_{in} - \omega_{laser} - i\gamma_{lt})} dt \tag{5.65}$$

From here on, it is possible to show that the $\alpha_{\rho\sigma}^{if}$ can be calculated as an integral of a correlation function [609], which is similar to the one previously discussed. In order to calculate that, a path integral scheme is also used and a geometry and Hessian for the ES are needed. The `ORCA_ESD` module predicts the ES PES (if not inputed), computes the α_{if} and then prints the Scattering factor on a spectrum named BASENAME.spectrum.LASERE.

OBS.: The actual Raman Intensity collected with any polarization at 90 degrees, the I($\pi/2; \|^s + \perp^s, \perp^i$ [610]), can be obtained by setting RRINTENS to TRUE under %ESD.

OBS2.: In the current implementation, if a multistate calculation is requested, Eq. (5.63) is solved for each state and all spectra are summed up afterwards.

Specific Keywords and Details

In order to solve Eq. (5.65), the same information as for Absorption/Emission is needed and to compute the ES PES all of the above approximations are also valid. The main difference here is that a laser energy, given by the LASERE flag, should be given. If it is not given, the default is to set it to the 0-0 energy difference between the ground and the excited state. As mentioned before, more information can be found on Sec. *Resonant Raman Spectrum*.

- You can choose more than one laser energy by setting LASERE 1,2,3,4 and etc. If so, each spectrum will be saved on a different BASENAME.spectrum.LASERE file.
- You can also choose more than one excited state to be accounted for with the flag STATES 1,2,3, etc. and the final spectrum will be the sum of all of Scattering Factors given by the α_{if}s. You can NOT choose several states and laser energies currently.
- The automatic selection for the integral grid is also done based on the same ideas as mentioned before.
- The default lineshape for resonant Raman is VOIGT.
- The lineshape of the RR spectra will be taken from the RRSLINEW flag. In this case, LINEW and INLINEW are used only in the calculation of the correlation function.
- Currently the only temperature for which this model works is at 0K.
- In terms of which vibrationally excited states to be considered, currently it goes up to Raman Order 2, which means fundamentals, first overtones and combination bands (up to a total quantum number of 2). You can reduced that using RORDER flag.

- It is also possible to include HT effect here for weak transitions, but be aware the calculation is much more costly. Due to technical reasons, the data is saved only on memory so, if you plan to go being 300 modes and do HT, there should be A LOT of memory available, about $8 \times NMODES^4$. Also, you should expect a VERY long time for the computation of the correlation function. We are currently working on ways to accelerate this particular case.
- As it is explained on the reference paper [609], the calculations using both Duschinsky rotation and HT effect can be greatly accelerated setting cutoffs for the derivatives and J matrix elements. The RRTCUTDER is a ratio with respect to the transition dipole moment below which the derivatives will be ignored and RRTCUTJ is a cutoff for J matrix elements. As the paper suggests, RRTCUTDER = 0.001 and RRTCUTJ = 0.01 are in general good numbers. We do recommend using these, but please be aware of the specific needs of your system.

5.5.12 Complete Keyword List for the ESD Module

```
%ESD                                #The booleans are the defaults
  ESDFLAG    ABS   (default)        #Which calculation to make?
             ECD
             FLUOR (default)
             CPF/CPFLUOR
             PHOSP
             CPP/CPPHOSP
             MCD                    #Only available with TDDFT
             RR
             ISC
             IC

  GSHESSIAN  "BASENAME.hess"        #The ground state Hessian
  ESHESSIAN  "BASENAME_S1.hess"     #The excited state Hessian
  TSHESSIAN  "BASENAME_T.hess"      #The triplet state Hessian

  HESSFLAG   AHAS                   #How to obtained the ES PES?
             VH
             VG (default)
             VGFC                   # VG + APPROXADEN TRUE
             HHBS
             HHAS
             UFBS
             UFAS

  STATES     1,2,3,4                #IROOTS to compute

  MODELIST   4,5,6                  # only include the 4th, 5th and 6th vibrational␣
↪modes
                                    # in the ESD calculation
  SINGLEMODE 1,5,10                 # run three ESD calculations, each considering␣
↪only
                                    # one of the 1st, 5th and 10th vibrational modes

  DOHT       FALSE                  #Do HT effect?
  FASTDER    TRUE                   #Use the fast derivatives algorithm?
  DELQ       0.01 (default)         #Dimensionless displacemente for derivatives
  STDA       FALSE                  #Use sTDA during derivatives?

  APPROXADEN FALSE                  #Compute the DELE by extrapolating info from
                                    #the Hessians, avoiding second single-point
                                    #at the ES geometry

  USEJ       FALSE                  #Consider Duschinsky rotations?
  USEB       TRUE                   #Rotate the initial state?
```

(continues on next page)

(continued from previous page)

```
  SAMEFREQ   TRUE                #Use DO method and J=1.

  DELE       12000               #Custom energy difference
  TDIP       x,y,z               #Custom transition electric dipole
             x.re,x.im,y.re,y.im,z.re,z.im
  TMDIP      x,y,z               #Custom transition magnetic dipole
             x.re,x.im,y.re,y.im,z.re,z.im
  SOCME      x.re,x.im           #Custom spin-orbit coupling matrix elements, in a.
↪u.
  LASERE     34000               #The laser energy for RR

  B          3000.0              #Magnetic field in Gauss for MCD
  LEBEDEVINTEGRATIONPOINTS 14    #Lebedev Grid for MCD molecular
                                 #orientational average

  GEOMSTEP   AUGHESS (default)   #Geometry step?
             QN
  STEPSCALING  1.0               #A number for scaling the steps
  STEPCONSTR   0,2,5             #A list of atoms that will not be moved
  COORDSYS   CART                #Coordinate system for the normal modes?
             INT (default)
             WINT
             FCWL
             FCWS

  TCUTFREQ   100                 #Cutoff for frequencies
  IFREQFLAG  POSITIVE (default)  #What to do with negative frequencies?
             LEAVE
             REMOVE

  UF_DELE    1E-4                #Energy difference for updated freq.
  UFFREQERR  0.2                 #Tolerated percentual error

  TEMP       298.15 (default)    #Temperature to consider
  UNIT       WN                  #wavenumbers  (Output units - input still in cm-1!
↪)
             NM                  #nanometers
             EV                  #electron volts
  CENTRALDIFF  TRUE              #Central differences?
  CONVDER    FALSE               #Convert derivatives between state
  SCALING    1.0 (default)       #Scaling for frequencies
  TDIPSCALING  1.0 (default)     #Scaling for the transition dipole

  LINES      DELTA               #The lineshape function
             LORENTZ (default)
             GAUSS
             VOIGT (default for RR)
  LINEW      50 (default)
  INLINEW    250 (default)
  RRSLINEW   10 (default)

  RORDER     2 (default)         #The Raman Order for RR
  RRINTENS   false               #Calculate the intensities instead
  RRTCUTDER  0 (default)         #A cutoff for derivatives
  RRTCUTJ    0 (default)         #A cutoff for J matrix elements

  WRITEHESS  FALSE               #Make ES PES and leave

  MAXTIME    12000               #Max time (atomic units!) for the FT
  NPOINTS    131072              #Total number of points
```

(continues on next page)

(continued from previous page)

```
  SPECRANGE  10000,50000         #Spectral range
  SPECRES    1.0                 #Spectral resolution

  PRINTVIB   FALSE               #Save a .xyz file per each vibational mode.
                                 #(Requires DOHT true)
  PRINTLEVEL 0                   #If set to 1 (or 2, 3, ...), prints additional
                                 #details of the calculation
END
```

5.6 Excited States via RPA, CIS, TD-DFT and SF-TDA

ORCA features a relatively efficient single-excitation CI (CIS), "random-phase approximation" (RPA) and time-dependent DFT module that can be used to calculate excitation energies, absorption intensities and CD intensities. Especially TD-DFT became very popular for excited state calculations as it offers significantly better results than HF-CIS at about the same cost. However, there are also many pitfalls of TD-DFT, some of which are discussed in reviews[626][627]. TD-DFT methods are available for closed-shell and spin-unrestricted reference states, together with its collinear spin-flip variant. Analytic gradients are available for all these cases. There also is a doubles correction implemented that improves the results (but also the computational cost). It is often used together with double-hybrid functionals as explained below. The TD-DFT module of ORCA is also extensively used for the calculation of X-ray absorption spectra at the K-edge of a given element.

Starting from version 6.0.0, the output format of the absorption wavelength, oscillator strength etc. has changed compared to the 5.0.x version. For more details on the interpretation of the output, please refer to *One Photon Spectroscopy*.

5.6.1 General Features

The module is invoked with the block:

```
%cis  end

# or equivalently

%tddft end
```

There are a variety of options. The most important one is the number of excited states that you want to have calculated:

```
%cis NRoots    10  end
```

The convergence tolerances are given by:

```
%cis
  ...
  ETol  1e-6
  RTol  1e-6
end
```

The variable `ETol` gives the required convergence of the energies of the excited states (in Eh) and `RTol` is the required convergence on the norm of the residual vectors. Under normal ciorcumstances the calculations need about 5-10 iterations to converge to the default convergence tolerances.

Once converged, the program prints the wave function composition. To keep the printing concise, coefficients smaller than 0.01 are omitted. The threshold can be adjusted with the keyword `TPrint`.

```
%cis
  ...
```

(continues on next page)

(continued from previous page)

```
  TPrint 0.0001 # cut-off for the wave function printing, default= 0.01
end
```

If closed-shell references are used the program can calculate the singlet and spin-adapted triplet excited states at the same time by using:

```
%cis
  ...
  triplets  true
end
```

This is available for all combinations of methods, including analytic gradients, and for double-hybrids.

In order to control the orbitals that should be taken into account in the calculation two mechanisms are available. The first mechanism is the default mechanism and consists of specifying and orbital energy window within which all single excitations will be considered:

```
%cis
  ...
  EWin  -3,3  # (orbital energy window in Eh)
end
```

Thus, the default is to keep core orbitals frozen and to neglect very high lying virtual orbitals which is a sensible approximation. However, you may want to consider to include all virtual orbitals by choosing for example `EWin -3,10000`. The second mechanism is to explicitly give an orbital energy window for each operator, i.e.

```
%cis
  ...
  OrbWin[0] = 2,-1,-1,14 # orbital window for spin-up MOs
  OrbWin[1] = 2,-1,-1,16 # orbital window for spin-down MOs
end
```

The "-1"'s in the above example mean that the HOMO and LUMO for the spin-.up and spin-down orbitals will be automatically determined by the program. In other words, in the above example, only the following excitations are included in the TDDFT calculation:

- Excitations from any occupied alpha orbital whose index is between 2 (inclusive) and that of the alpha HOMO (inclusive), to any virtual alpha orbital whose index is between that of alpha LUMO (inclusive) and 14 (inclusive)
- Excitations from any occupied beta orbital whose index is between 2 (inclusive) and that of the beta HOMO (inclusive), to any virtual beta orbital whose index is between that of beta LUMO (inclusive) and 16 (inclusive)

For calculations based on a restricted reference, `OrbWin[1]` will be ignored.

In using the CIS/TD-DFT module five different types of calculations should be distinguished:

- Semiempirical methods
- Hartree-Fock calculations
- DFT calculations without HF exchange (non-hybrid functionals)
- DFT calculations with HF exchange (hybrid functionals)
- DFT calculations with HF exchange and MP2 correlation (double-hybrid functionals)

5.6.2 Semiempirical Methods

The semiempirical INDO/S method is very suitable to calculate absorption spectra of medium sized to large organic and inorganic molecules. It has been parameterized by the late M. C. Zerner for optical spectroscopy and in my experience at least, it tends to work nicely for many systems. With the semiempirical approach it is easy to calculate many states of large molecules. For example, consider the following calculation on a bis-histidine ligated iron-porphyrin model (in the Fe(II) state) that includes 92 atoms and ≈ 16,500 CSFs in the single excitation space. Yet the calculation requires only a few minutes on an ordinary computer for the prediction of the first 40 excited states.

The calculated spectrum is in essentially reasonable agreement with experiment in showing a huge band around 400 nm (the famous Soret band) and a smaller but still intense band between 500 and 550 nm (the Q-band). There are no predicted absorptions below ≈ 10,000 cm^{-1}.

The input for the job is shown below:

```
# Test CIS in conjunction with INDO/S

! ZINDO/S TightSCF DIIS NoMOPrint
%cis  NRoots  40
      end
* xyz 0 1
Fe   -0.01736     0.71832    -0.30714
C     2.65779     4.03195    -0.13175
C     3.51572     3.02488    -0.24101
C     2.66971     1.82027    -0.30891
C     3.30062     0.51609    -0.42755
C     2.61022    -0.60434    -0.47131
C     3.32146    -1.89491    -0.57434
C     2.35504    -2.79836    -0.57179
C     1.11740    -1.99868    -0.46878
C    -0.04908    -2.61205    -0.44672
C    -1.30967    -1.89127    -0.38984
C    -2.58423    -2.63345    -0.40868
C    -3.50492    -1.68283    -0.37930
C    -2.72946    -0.42418    -0.33711
C    -3.35747     0.73319    -0.28970
C    -2.66935     2.01561    -0.22869
C    -3.31167     3.19745    -0.16277
C    -4.72835     3.62642    -0.14517
C    -5.84825     2.89828    -0.20597
C    -2.21443     4.15731    -0.09763
C    -1.11572     3.39398    -0.14235
C     0.19578     4.02696    -0.10122
C     1.33370     3.36290    -0.15370
C     3.09165     5.44413    -0.02579
C     2.35656     6.55323     0.10940
N     1.43216     2.09428    -0.24815
N     1.34670    -0.74673    -0.42368
N    -1.39885     2.15649    -0.21891
N    -1.47620    -0.63353    -0.34705
C     5.03025     3.02708    -0.28544
C     4.81527    -2.12157    -0.66646
C    -5.01065    -1.83771    -0.38886
C    -2.28137     5.66820    -0.00321
C    -2.73691    -4.14249    -0.43699
C    -2.42579    -4.72805    -1.83259
C     2.45978    -4.31073    -0.64869
C     2.19678    -4.82182    -2.08201
C     1.60835    -6.22722    -2.10748
C    -1.90102    -6.15737    -1.82447
O    -1.96736    -6.92519    -2.75599
O     1.60982    -7.01844    -1.19330
```

(continues on next page)

(continued from previous page)

```
O    -1.15355    -6.41323    -0.74427
O     0.89871    -6.41433    -3.22828
H     4.17823     5.62170    -0.05623
H     2.86221     7.53117     0.17503
H     1.26303     6.57673     0.17212
H     0.21799     5.11603    -0.03468
H    -1.78003     6.14426    -0.87498
H    -3.32281     6.05139     0.01906
H    -1.78374     6.03115     0.92347
H    -4.89690     4.71221    -0.07358
H    -6.82566     3.40843    -0.18007
H    -5.88239     1.80643    -0.28628
H    -4.44893     0.70720    -0.28575
H    -5.32107    -2.89387    -0.54251
H    -5.45075    -1.49552     0.57400
H    -5.46788    -1.24144    -1.20929
H    -2.05997    -4.55939     0.34045
H    -3.76430    -4.43895    -0.12880
H    -3.33638    -4.66246    -2.47119
H    -1.65517    -4.10119    -2.33605
H    -0.56422    -7.14866    -1.00437
H     0.26056    -7.12181    -3.00953
H     1.48118    -4.13253    -2.58671
H     3.13949    -4.79028    -2.67491
H     3.46153    -4.65168    -0.30336
H     1.73023    -4.75206     0.06633
H     5.26172    -1.51540    -1.48550
H     5.31767    -1.84036     0.28550
H     5.06416    -3.18438    -0.87628
H    -0.07991    -3.70928    -0.48866
H     4.39835     0.46775    -0.47078
H     5.39550     2.59422    -1.24309
H     5.47197     4.04179    -0.19892
H     5.44914     2.41988     0.54738
N     0.01831     0.60829     1.68951
C     0.02054     1.64472     2.54371
C     0.04593    -0.50152     2.45186
N     0.04934     1.20474     3.84418
C     0.06582    -0.16578     3.80848
H     0.00322     2.72212     2.31829
N    -0.05051     0.81937    -2.30431
H     0.05251    -1.53704     2.08183
C     0.11803     1.92670    -3.04495
H     0.05712     1.81091     4.70485
H     0.08982    -0.83278     4.68627
C    -0.24302    -0.18840    -3.17641
C    -0.19749     0.28568    -4.49059
N     0.03407     1.63309    -4.38373
H     0.30109     2.95786    -2.70479
H    -0.41432    -1.24242    -2.91290
H    -0.31761    -0.27403    -5.43315
H     0.12975     2.31943    -5.17616
*
```

Fig. 5.18: Structure of the iron-porphyrin used for the prediction of its absorption spectrum (the structure was obtained from a molecular mechanics calculation and the iron-imidazole bondlength was set to 2.0 Å).

Fig. 5.19: The ZINDO/S predicted absorption spectrum of the model iron porphyrin shown above. The spectrum has been plotted using the `orca_mapspc` tool.

> **Note**
>
> ORCA slightly departs from standard ZINDO/S in using dipole integrals in the intensity calculations that include all one- and two-center terms which are calculated via a STO-3G expansion of the Slater basis orbitals. The calculated intensities are not highly accurate anyways. In the present case they are overestimated by a factor of $\approx$ 2.

5.6.3 Hartree-Fock Wavefunctions

When applying the procedures outlined above to pure Hartree-Fock, one obtains the "random-phase approximation" (RPA) or the CI singles (CIS) model (when effectively using the Tamm-Dancoff Approximation, TDA). In general, RPA and CIS calculations do not lead to good agreement with experimental excitation energies and errors of 1-5 eV are common. Therefore HF/CIS is mostly a qualitative tool or can be used with caution for larger molecules if more extensive and more well balanced CI calculations are not computationally tractable.

5.6.4 Non-Hybrid and Hybrid DFT

For DFT functionals there is the choice between the full TD-DFT (eq. (5.66)) treatment and the so-called Tamm-Dancoff approximation (TDA).

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B*} & \mathbf{A*} \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = \begin{pmatrix} \omega & 0 \\ 0 & -\omega \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} \tag{5.66}$$

The TDA is the same approximation that leads from RPA to CIS (i.e. neglect of the so-called "B" matrix, see eq. (5.67)). The results for vertical excitation energies are usually very similar between the two approaches.

$$\mathbf{A}\mathbf{X}_{\text{TDA}} = \omega_{\text{TDA}}\mathbf{X}_{\text{TDA}} \tag{5.67}$$

In general, the elements of matrix "A" and "B" for singlet-singlet excitations in the spin-restricted case are given by eqs. (5.68) and (5.69).

$$\begin{aligned} A_{ia,jb} =& \delta_{ij}\delta_{ab}(\epsilon_a - \epsilon_i) + 2(ia|jb) - a_{\text{X}}(ij|ab) \\ &+ (1 - a_{\text{X}})(ia|f_{\text{XC}}|jb) \end{aligned} \tag{5.68}$$

and

$$B_{ia,jb} = 2(ia|bj) - a_{\text{X}}(ib|aj) + (1 - a_x)(ia|f_{\text{XC}}|bj). \tag{5.69}$$

Here, i, j denote occupied and a, b virtual orbitals. a_{X} is the amount of non-local Fock exchange in the density functional. If a_{X} is equal to one, eqs. (5.66) and (5.67) correspond to the RPA and CIS case, based on a Hartree-Fock ground state determinant.

The TDA is actually the default method for TD-DFT, and can be turned off by:

```
%tddft
  TDA  false
end
```

There are situations where hybrid functionals give significantly better results than pure functionals since they suffer less from the self-interaction error. In those cases, the RIJCOSX procedure[119] [628][120] leads to very large speedups in such calculations at virtually no loss in accuracy[629], and is turned on by default whenever the SCF uses that too.

5.6.5 Collinear Spin-Flip TDA (SF-TD-DFT)

Another approach to obtain excited states via CIS/TD-DFT are the so called spin-flip methods (for a good review, please check ref [630]). The idea is to start from an UHF state, and then "flip" one of the alpha electrons to generate states with $MS_{SF} = MS_{UHF} - 1$. In order to do that, we look for excitations from alpha-to-beta orbitals only, and that makes the A matrix from TDA even simpler:

$$A^{SF}_{i\bar{a},j\bar{b}} = \delta_{ij}\delta_{\bar{a}\bar{b}}(\epsilon_{\bar{a}} - \epsilon_i) - a_{\text{X}}(ij|\bar{a}\bar{b}) \tag{5.70}$$

where the overbar represent beta orbitals, and no-overbars alpha orbitals.

> **Note**
>
> Please note that for pure DFT (with $a_X = 0$, and no HF contribution), the A matrix is based simply in the orbital energies, and thus it is always good to have a good amount of HF on the functional!

In order to facilitate the discussion on the results one gets from the SF-TDA, let's take a closer look at the picture representing some possible excitations:

Fig. 5.20: Effect of the spin-flip operator on a UHF ($MS = 3$) wavefunction. The "spin-complete" states are eigenvectors of the S^2 operator, while the "spin-incomplete" are not. Alpha and beta orbitals here are represented with the same energy, just to simplify the image. Adapted from the previously mentioned review.

It is important to note that no all SF-excitations lead to determinants that are eigenvalues of the S^2 operator. That means, depending on how much of these "spin incomplete" excitations are present in the final SF-state, the spin-contamination could be high, and in this case, states with $\langle S^2 \rangle \simeq 1$ would be predicted. These are undefined states within the SF theory and should be treated carefully.

Important

- Any SF method can only be used starting from a UHF wavefunction, with a multiplicity of at least 3!

First example: methylene and SF-CIS

One simple example is the calculation of the vertical singlet-triplet splitting of the methylene radical within CIS, using the following input with symmetry included:

```
!def2-SVP USESYM

%TDDFT
 SF TRUE
END

* xyz 0 3
 C 0  0       0.1058
 H 0  0.9910 -0.3174
 H 0 -0.9910 -0.3174
*
```

The geometry was taken from a high-level CCSD(T)/cc-pVQZ (X^3B_1) optimized geometry, and after the regular UHF SCF, the SF-CIS result is:

```
---------------------
SF-CIS EXCITED STATES
---------------------

the weight of the individual excitations are printed if larger than 1.0e-02

UHF/UKS reference: multiplicity estimated based on rounded <S**2> value, RELEVANCE␣
↪IS LIMITED!
```

(continues on next page)

(continued from previous page)

```
(SPIN-FLIP GROUND STATE)
STATE  1:  E=   0.005456 au      0.148 eV     1197.5 cm**-1 <S**2> =   2.041850␣
↪Sym: B1 Mult 3
     1a ->   3b  :     0.017437 (c= -0.13204740)
     3a ->   3b  :     0.465217 (c=  0.68206845)
     3a ->   9b  :     0.026214 (c=  0.16190881)
     4a ->   4b  :     0.443967 (c= -0.66630872)
     4a ->  10b  :     0.028255 (c= -0.16809333)

STATE  2:  E=   0.058434 au      1.590 eV    12824.9 cm**-1 <S**2> =   0.022791␣
↪Sym: A1 Mult 1
     3a ->   4b  :     0.128149 (c= -0.35797971)
     4a ->   3b  :     0.810350 (c=  0.90019420)
     4a ->   9b  :     0.032289 (c=  0.17969015)

STATE  3:  E=   0.079264 au      2.157 eV    17396.4 cm**-1 <S**2> =   0.032278␣
↪Sym: B1 Mult 1
     3a ->   3b  :     0.446020 (c= -0.66784709)
     3a ->   9b  :     0.018620 (c= -0.13645512)
     4a ->   4b  :     0.494589 (c= -0.70327031)
     4a ->  10b  :     0.023225 (c= -0.15239893)
```

Now, **it is very important to consider that the SF ground state is not the UHF ground state anymore**, the "new" ground state within the SF scheme is actually STATE 1. You can think of the UHF as being only an initial model, on the basis of which the SF states are built. The final energy of the new ground state is actually the SCF energy + energy of the STATE 1 (which is the one given as the FINAL SINGLE POINT ENERGY is no IROOT is given). This last contribution can be either positive or negative, depending on the case.

Anyway, the ground state is predicted to be a triplet state (here with $M_S = 0$), as expected for this carbene, and the S-T spiting energy is 1.590 - 0.148 eV = 1.442 eV. The full CI results for that is 1.50 eV, so it is already almost there! Of course, in this case computing the RHF singlet - UHF triplet makes no sense, since the RHF singlet would not have the necessary open-shell singlet character.

Benzyne and SF-TDA

Benzyne is a classic diradical that can be generated from benzene by hydrogen abstraction (Fig. 5.21). It is known to have an open-shell singlet ground state, and has its adiabatic sinlget-triplet splitting measured experimentally. Let's try to compute this value using SF-TDA with ORCA.

Fig. 5.21: Lewis representation of the benzene and benzyne molecules, indicating the diradical character of the later.

First, we optimize the open-shell singlet by using SF, and the input that follows. Here we use now DFT, in particular the BHANDHLYP functional, which uses 50% of HF correlation, and is recommended for this kind of application. By default, the IROOT to be optimized is 1, which in this case corresponds to the SF ground state.

```
!DEF2-TZVPD OPT

%method
 Method DFT
 Functional hyb_gga_xc_bhandhlyp
end
```

(continues on next page)

(continued from previous page)

```
%tddft
  SF true
  NRoots  3
end

* xyz 0 3
  C         -1.39113        0.00000        0.00000
  C          0.69557        1.20476        0.00000
  C         -0.69557        1.20476        0.00000
  C         -0.69557       -1.20476        0.00000
  C          0.69557       -1.20476        0.00000
  C          1.39113        0.00000        0.00000
  H         -1.24291        2.15278        0.00000
  H         -1.24291       -2.15278        0.00000
  H          1.24291       -2.15278        0.00000
  H          1.24291        2.15278        0.00000
*
```

And after the optimization of IROOT 1, the final SF-TDA result is:

```
---------------------
SF-TDA EXCITED STATES
---------------------

the weight of the individual excitations are printed if larger than 1.0e-02

UHF/UKS reference: multiplicity estimated based on rounded <S**2> value, RELEVANCE␣
↪IS LIMITED!

(SPIN-FLIP GROUND STATE)
STATE  1:  E=   0.024173 au      0.658 eV     5305.3 cm**-1 <S**2> =   0.023478␣
↪Mult 1
    11a ->  19b  :     0.018563 (c= -0.13624537)
    17a ->  20b  :     0.245232 (c=  0.49520915)
    17a ->  27b  :     0.016815 (c= -0.12967111)
    20a ->  19b  :     0.667236 (c=  0.81684517)
    20a ->  25b  :     0.019935 (c= -0.14119296)

STATE  2:  E=   0.032615 au      0.887 eV     7158.1 cm**-1 <S**2> =   2.018056␣
↪Mult 3
    11a ->  20b  :     0.015462 (c= -0.12434444)
    17a ->  19b  :     0.448809 (c=  0.66993209)
    17a ->  25b  :     0.017854 (c= -0.13361796)
    20a ->  20b  :     0.460923 (c=  0.67891314)
    20a ->  27b  :     0.024031 (c= -0.15501804)

STATE  3:  E=   0.106201 au      2.890 eV    23308.4 cm**-1 <S**2> =   1.029543␣
↪Mult 3
    15a ->  20b  :     0.051523 (c= -0.22698783)
    18a ->  19b  :     0.910633 (c= -0.95427103)
    18a ->  25b  :     0.017352 (c=  0.13172868)
```

confirming the singlet ground state, with an upper triplet excited state.

Now to optimize the triplet state using SF-TDA, one has to use a similar input, except that now IROOT 2 has to be chosen as the one to be optimized:

```
!DEF2-TZVPD OPT

%method
  Method DFT
```

(continues on next page)

(continued from previous page)

```
  Functional hyb_gga_xc_bhandhlyp
end

%tddft
  SF true
  NRoots  3
  IRoot   2
end

* xyz 0 3
  C         -1.39113        0.00000        0.00000
  C          0.69557        1.20476        0.00000
  C         -0.69557        1.20476        0.00000
  C         -0.69557       -1.20476        0.00000
  C          0.69557       -1.20476        0.00000
  C          1.39113        0.00000        0.00000
  H         -1.24291        2.15278        0.00000
  H         -1.24291       -2.15278        0.00000
  H          1.24291       -2.15278        0.00000
  H          1.24291        2.15278        0.00000
*
```

Giving the following results:

```
---------------------
SF-TDA EXCITED STATES
---------------------

the weight of the individual excitations are printed if larger than 1.0e-02

UHF/UKS reference: multiplicity estimated based on rounded <S**2> value, RELEVANCE␣
↪IS LIMITED!

(SPIN-FLIP GROUND STATE)
STATE  1:  E=   0.028669 au      0.780 eV     6292.1 cm**-1 <S**2> =   0.022484␣
↪Mult 1
    11a ->  19b  :     0.018011 (c= -0.13420632)
    17a ->  20b  :     0.293971 (c=  0.54219090)
    17a ->  27b  :     0.017459 (c= -0.13213220)
    20a ->  19b  :     0.611212 (c=  0.78180024)
    20a ->  25b  :     0.023243 (c=  0.15245717)

STATE  2:  E=   0.032738 au      0.891 eV     7185.1 cm**-1 <S**2> =   2.018181␣
↪Mult 3
    11a ->  20b  :     0.014963 (c= -0.12232498)
    17a ->  19b  :     0.444520 (c=  0.66672364)
    17a ->  25b  :     0.021443 (c=  0.14643407)
    20a ->  20b  :     0.462361 (c=  0.67997124)
    20a ->  27b  :     0.022005 (c= -0.14834057)

STATE  3:  E=   0.108154 au      2.943 eV    23737.1 cm**-1 <S**2> =   1.030736␣
↪Mult 3
    15a ->  20b  :     0.052291 (c= -0.22867148)
    18a ->  19b  :     0.904505 (c= -0.95105480)
    18a ->  25b  :     0.021057 (c= -0.14511042)
```

After the optimization, the final predicted adiabatic singlet-triplet gap is 0.233 eV, compared to the experimental value of 0.165 eV [631].

5.6.6 Including solvation effects via LR-CPCM theory

The LR-CPCM theory, as developed by Cammi and Tomasi [632], is implemented for both energies and gradients of excited states. It is turned on by default, whenever CPCM is also requested for the ground state.

The major change is that now there is a $G_{ia,jb}$ term in the **A** part of Eq. (5.66), related to solvation effects.

$$\begin{aligned} A_{ia,jb} =& \delta_{ij}\delta_{ab}(\epsilon_a - \epsilon_i) + 2(ia|jb) + 2G_{ia,jb} \\ & - a_{\mathrm{X}}(ij|ab) + (1 - a_{\mathrm{X}})(ia|f_{\mathrm{XC}}|jb) \end{aligned} \tag{5.71}$$

where $G_{ia,bj}$ is defined as:

$$G_{ia,jb} = (\mathbf{V}_{ia})^T \mathbf{q}_{jb} \tag{5.72}$$

Equilibrium and non-equilibrium conditions

These charges q_{jb} are calculated in the same way as described in *The Conductor-like Polarizable Continuum Model (C-PCM)*, but for excited states, two different values of ε can be used, depending on the dynamics of the system:

- **Non-equilibrium**: If the calculation assumes that the electronic excitation is so fast, that there is no time for the solvent to reorganize around the solute, then the $\varepsilon_{\mathrm{inf}}$ of the solvent is used, which is equivalent to the square of the refractive index. That is the case if one wants to compute the vertical excitation energy, and it is the default in that case.
- **Equilibrium**: If the excited state is assumed to be completely solvated, then the true dielectric constant ε of the solvent should be used. That is the case for geometry optimizations, frequencies or inside ORCA_ESD. This is turned on by default whenever analytic gradients are requested.

In any case, these conditions can be controlled by the flag CPCMEQ, that can be set to TRUE or FALSE by the user, and will then override the defaults.

These are available to all CIS/TD-DFT options: singlets, spin-adapted triplets, UHF and spin-flip variants. It works inclusive for double-hybrids and whenever SOC is requested.

5.6.7 Population Analysis of Excited States

If you want to print a population analysis for the excited state using CIS/TD-DFT, there are two options available: using **unrelaxed** or **relaxed** densities. For the unrelaxed densities, simply use UPOP TRUE:

```
!PBE def2-SVP

%tddft
  NROOTS  5
  UPOP    TRUE
end

* xyz 0 1
  O         -1.88199        1.42016       -0.00000
  C         -1.80947        0.20286        0.00000
  H         -2.50488       -0.38174       -0.59212
  H         -1.04956       -0.29504        0.59212
*
```

and the atomic changes and bond orders will be printed for the chosen IROOT (default 1):

```
-----------------------------------------------------------------------------
                 UNRELAXED CIS/TDA DENSITY POPULATION ANALYSIS
                                   IROOT 1
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
                           ORCA POPULATION ANALYSIS
-------------------------------------------------------------------------------
Input electron density              ... example5.cispur.singlet.iroot0
BaseName (.gbw .S,...)              ... example5

                     ********************************
                     * MULLIKEN POPULATION ANALYSIS *
                     ********************************

-----------------------
MULLIKEN ATOMIC CHARGES
-----------------------
   0 O :    0.177629
   1 C :   -0.438661
   2 H :    0.130515
   3 H :    0.130518
Sum of atomic charges:   -0.0000000

(...)

                       ******************************
                       * MAYER POPULATION ANALYSIS *
                       ******************************

  NA   - Mulliken gross atomic population
  ZA   - Total nuclear charge
  QA   - Mulliken gross atomic charge
  VA   - Mayer's total valence
  BVA  - Mayer's bonded valence
  FA   - Mayer's free valence

  ATOM       NA         ZA         QA         VA         BVA        FA
  0 O      7.8224     8.0000     0.1776     2.5554     1.4462     1.1092
  1 C      6.4387     6.0000    -0.4387     3.8364     3.1848     0.6516
  2 H      0.8695     1.0000     0.1305     0.9720     0.8521     0.1199
  3 H      0.8695     1.0000     0.1305     0.9720     0.8521     0.1199

  Mayer bond orders larger than 0.100000
B(  0-O ,  1-C ) :   1.4477 B(  1-C ,  2-H ) :   0.8685 B(  1-C ,  3-H ) :   0.8685
```

To get the analysis from the relaxed density, simply use `!ENGRAD` to a run a gradient calculation:

```
!PBE def2-SVP ENGRAD

%tddft
  NROOTS  5
  UPOP    TRUE
end

* xyz 0 1
  O         -1.88199        1.42016       -0.00000
  C         -1.80947        0.20286        0.00000
  H         -2.50488       -0.38174       -0.59212
  H         -1.04956       -0.29504        0.59212
*
```

and the printout is:

```
-------------------------------------------------------------------------------
                  RELAXED CIS/TDA DENSITY POPULATION ANALYSIS
                                   IROOT 1
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
                           ORCA POPULATION ANALYSIS
--------------------------------------------------------------------------------
Input electron density              ... example6.cispre.singlet.iroot1
BaseName (.gbw .S,...)              ... example6

                     *******************************
                     * MULLIKEN POPULATION ANALYSIS *
                     *******************************

-----------------------
MULLIKEN ATOMIC CHARGES
-----------------------
   0 O :   -0.092919
   1 C :   -0.088136
   2 H :    0.090533
   3 H :    0.090522
Sum of atomic charges:   -0.0000000

(...)

                       ******************************
                       * MAYER POPULATION ANALYSIS *
                       ******************************

  NA   - Mulliken gross atomic population
  ZA   - Total nuclear charge
  QA   - Mulliken gross atomic charge
  VA   - Mayer's total valence
  BVA  - Mayer's bonded valence
  FA   - Mayer's free valence

  ATOM       NA         ZA         QA         VA         BVA        FA
  0 O      8.0929     8.0000    -0.0929     2.2728     1.3646     0.9082
  1 C      6.0881     6.0000    -0.0881     3.9322     3.1290     0.8031
  2 H      0.9095     1.0000     0.0905     0.9876     0.8843     0.1034
  3 H      0.9095     1.0000     0.0905     0.9876     0.8843     0.1034

  Mayer bond orders larger than 0.100000
B(  0-O ,  1-C ) :   1.3485 B(  1-C ,  2-H ) :   0.8902 B(  1-C ,  3-H ) :   0.8902
```

In order to print the analysis for multiple states, simply use `IROOTLIST` and `TROOTLIST`:

```
!PBE def2-SVP

%tddft
  NROOTS  5
  IROOTLIST 1,2,3
  TROOTLIST 1,2,3
  UPOP    TRUE
end

* xyz 0 1
  O         -1.88199        1.42016       -0.00000
  C         -1.80947        0.20286        0.00000
  H         -2.50488       -0.38174       -0.59212
  H         -1.04956       -0.29504        0.59212
*
```

5.6.8 Simplified TDA and TD-DFT

ORCA also supports calculations of excited states using the simplified Tamm-Dancoff approach (sTDA) by S. Grimme[633]. The sTDA is particularly suited to calculate absorption spectra of very large systems. sTDA as well as the simplified time-dependent density functional theory (sTD-DFT)[634] approach require a (hybrid) DFT ground state calculation. For large systems, using range-separated hybrid functionals (e.g. ωB97X) is recommended.[635] The sTD-DFT approach in particular yields much better electronic circular dichroism (ECD) spectra and should be used for this purpose.

Theoretical Background

A brief outline of the theory will be given in the following. For more details, please refer to the original papers[633, 634]. In the sTDA, the TDA eigenvalue problem from eq. (5.67) is solved using a truncated and semi-empirically simplified A' matrix. The trunctation negelects all excitations that are beyond the energy range of interest, except a few strongly coupled ones. The matrix elements from eq. (5.68) are simplified by neglecting the response of the density functional and by approximating the remaining two-electron integrals as damped Coulomb interactions between transition/charge density monopoles. In the following, the indices i, j denote occupied, a, b virtual and p, q either kind of orbitals.

$$A'_{ia,jb} = \delta_{ij}\delta_{ab}(\epsilon_a - \epsilon_i) + \sum_{A,B}^{N_{\text{atoms}}} (2q^A_{ia}\gamma^K_{AB}q^B_{jb} - q^A_{ij}\gamma^J_{AB}q^B_{ab}) \tag{5.73}$$

q^A_{pq} and q^B_{pq} are the transition/charge density monopoles located on atom A and B, respectively. These are obtained from Löwdin population analysis (see Sec. *Löwdin Population Analysis*). ϵ_p is the Kohn-Sham orbital energy of orbital p. γ^K_{AB} and γ^J_{AB} are the Mataga-Nishimoto-Ohno-Klopman damped Coulomb operators for exchange-type (K) and Coulomb-type (J) integrals, respectively.

$$\gamma^J_{AB} = \left(\frac{1}{(R_{AB})^\beta + (a_X\eta)^{-\beta}}\right)^{\frac{1}{\beta}} \tag{5.74}$$

$$\gamma^K_{AB} = \left(\frac{1}{(R_{AB})^\alpha + \eta^{-\alpha}}\right)^{\frac{1}{\alpha}} \tag{5.75}$$

Here, η is the arithmetic mean of the chemical hardness of atom A and B. α and β are the parameters of the method and are given by:

$$\alpha = \alpha_1 + a_x\alpha_2 \tag{5.76}$$

$$\beta = \beta_1 + a_x\beta_2 \tag{5.77}$$

For any global hybrid functional, α_1, α_2, β_1 and β_2 are identical. α and β then depend on the amount of Fock exchange (a_X) only. This is different for range-separated hybrid functionals where α_2 and β_2 are set to zero. α_1 and β_1 along with a value a_x for the sTDA treatment are individually fitted for each range-separated hybrid functional.[635] It can bee seen from eq. (5.73) that the method is asymptotically correct which is crucial for excitations of charge transfer type.

In sTD-DFT, eq. (5.66) is solved using the simplified matrices A' (see above) and B'.

$$B'_{ia,jb} = \sum_{A,B}^{N_{\text{atoms}}} (2q^A_{ia}\gamma^K_{AB}q^B_{bj} - a_X q^A_{ib}\gamma^K_{AB}q^B_{aj}) \tag{5.78}$$

This approach yields better transition dipole moments and therefore spectra but the method is more costly than sTDA (a factor of 2–5 for typical systems). The parameters used in sTDA and sTD-DFT are identical. There are **no** additional parameters fitted for this method.

Calculation Set-up

sTDA and sTD-DFT can be combined with any (restricted or unrestricted) hybrid DFT singlepoint calculation. Gradients and frequencies are **not** implemented! The methods can be invoked via the `%tddft` block. Table *Keyword list for sTDA and sTD-DFT.* gives a list of the possible keywords.

Table 5.6: Keyword list for sTDA and sTD-DFT.

Mode sTDA	Invokes a sTDA calculation
Mode sTDDFT	Invokes a sTD-DFT calculation
EThresh *value*	Energy threshold up to which CSFs are included (in eV)
PTLimit *value*	Energy threshold up to which CSFs beyond EThresh may be selected (in eV)
PThresh *value*	Selection criterion to include CSF beyond EThresh (in Eh)
axstda *value*	Fock exchange parameter used in sTDA/sTD-DFT calculation (for range-separated hybrids)
beta1 *value*	Constant part of J integral parameter β
beta2 *value*	a_X scaled part of J integral parameter β
alpha1 *value*	Constant part of K integral parameter α
alpha2 *value*	a_X scaled part of K integral parameter α
triplets true	Calculate singlet-triplet excitations (default: singlet-singlet)

The following example shows how to run such a sTDA calculation using the BHLYP functional if one is interested in all excitations up to 10 eV.

```
! bhlyp def2-SVP tightscf smallprint printgap nopop

%tddft
 Mode sTDA
 Ethresh 10.0
end

*xyz 0 1
  N     -1.726176    0.687742   -0.007335
  N      1.213763    0.735949   -0.027163
  N     -1.545188   -2.243058   -0.016803
  N      1.395005   -2.195166   -0.036972
  C      2.736733   -1.895646   -0.045209
  C      3.277248   -0.608637   -0.045175
  C      2.575908    0.605756   -0.036945
  C      0.977551    2.083559   -0.021690
  C     -0.288108    2.687287   -0.011220
  C     -1.529488    2.048274   -0.004419
  C     -3.067756    0.388045    0.001354
  H     -0.976948   -0.007843   -0.015911
  C     -3.608477   -0.899096    0.001258
  C     -2.907149   -2.113434   -0.007085
  C     -1.308388   -3.590806   -0.022331
  C     -0.042695   -4.194596   -0.032673
  C      1.198590   -3.555847   -0.039419
  H      0.645751   -1.499625   -0.028098
  C     -3.768181    1.641701    0.010433
  C     -2.832985    2.651061    0.006924
  C      2.228928    2.835679   -0.028158
  C      3.228183    1.912006   -0.037679
  C      3.437220   -3.149227   -0.053963
  C      2.502048   -4.158627   -0.050442
  C     -3.558669   -3.420017   -0.006390
  C     -2.559381   -4.343652   -0.015901
  H     -4.633051   -3.589818    0.000518
```

(continues on next page)

(continued from previous page)

```
  H     -2.644718   -5.427990   -0.018393
  H     -0.016400   -5.285446   -0.035842
  H      2.681678   -5.229875   -0.055110
  H      4.519010   -3.247119   -0.061999
  H      4.366739   -0.548807   -0.052422
  H      2.314239    3.920088   -0.025633
  H      4.302487    2.081864   -0.044563
  H     -0.315062    3.778120   -0.007889
  H     -3.013895    3.722099    0.011827
  H     -4.849852    1.740964    0.018679
  H     -4.697992   -0.958441    0.008649
end
```

Replacing `Mode sTDA` by `Mode sTDDFT` will invoke a sTD-DFT calculation instead. This is shown in the next example in combination with the ωB97X functional and user specified parameters:

```
! wb97x def2-SVP tightscf smallprint printgap nopop

%tddft
 Mode sTDDFT
 Ethresh 10.0
 axstda  0.56
 beta1  8.00
 beta2  0.00
 alpha1 4.58
 alpha2 0.00
end

*xyz 0 1
  N     -1.726176    0.687742   -0.007335
  N      1.213763    0.735949   -0.027163
  N     -1.545188   -2.243058   -0.016803
  N      1.395005   -2.195166   -0.036972
  C      2.736733   -1.895646   -0.045209
  C      3.277248   -0.608637   -0.045175
  C      2.575908    0.605756   -0.036945
  C      0.977551    2.083559   -0.021690
  C     -0.288108    2.687287   -0.011220
  C     -1.529488    2.048274   -0.004419
  C     -3.067756    0.388045    0.001354
  H     -0.976948   -0.007843   -0.015911
  C     -3.608477   -0.899096    0.001258
  C     -2.907149   -2.113434   -0.007085
  C     -1.308388   -3.590806   -0.022331
  C     -0.042695   -4.194596   -0.032673
  C      1.198590   -3.555847   -0.039419
  H      0.645751   -1.499625   -0.028098
  C     -3.768181    1.641701    0.010433
  C     -2.832985    2.651061    0.006924
  C      2.228928    2.835679   -0.028158
  C      3.228183    1.912006   -0.037679
  C      3.437220   -3.149227   -0.053963
  C      2.502048   -4.158627   -0.050442
  C     -3.558669   -3.420017   -0.006390
  C     -2.559381   -4.343652   -0.015901
  H     -4.633051   -3.589818    0.000518
  H     -2.644718   -5.427990   -0.018393
  H     -0.016400   -5.285446   -0.035842
  H      2.681678   -5.229875   -0.055110
  H      4.519010   -3.247119   -0.061999
  H      4.366739   -0.548807   -0.052422
```

(continues on next page)

(continued from previous page)

```
  H      2.314239    3.920088   -0.025633
  H      4.302487    2.081864   -0.044563
  H     -0.315062    3.778120   -0.007889
  H     -3.013895    3.722099    0.011827
  H     -4.849852    1.740964    0.018679
  H     -4.697992   -0.958441    0.008649
end
```

For the range-separated hybrid functionals LC-BLYP, CAM-B3LYP, ωB97, ωB97X, ωB97X-D3 and ωB97X-D3BJ, parameters are available and will be used by default if one of these functionals is used. The way of specifying parameters as shown above is useful if there is a range-separated hybrid functional that has not been parametrized for sTDA yet. For very large systems (e.g. $>$ 500 atoms), it may be useful to define an upper boundary `PTLimit` for the selection of configurations that are beyond `EThresh` (otherwise the whole configuration space will be scanned). This can be done as shown below:

```
! cam-b3lyp def2-SVP tightscf smallprint printgap nopop

%tddft
 Mode sTDDFT
 Ethresh 10.0
 PThresh 1e-4
 PTLimit 30
end

*xyz 0 1
  N     -1.726176    0.687742   -0.007335
  N      1.213763    0.735949   -0.027163
  N     -1.545188   -2.243058   -0.016803
  N      1.395005   -2.195166   -0.036972
  C      2.736733   -1.895646   -0.045209
  C      3.277248   -0.608637   -0.045175
  C      2.575908    0.605756   -0.036945
  C      0.977551    2.083559   -0.021690
  C     -0.288108    2.687287   -0.011220
  C     -1.529488    2.048274   -0.004419
  C     -3.067756    0.388045    0.001354
  H     -0.976948   -0.007843   -0.015911
  C     -3.608477   -0.899096    0.001258
  C     -2.907149   -2.113434   -0.007085
  C     -1.308388   -3.590806   -0.022331
  C     -0.042695   -4.194596   -0.032673
  C      1.198590   -3.555847   -0.039419
  H      0.645751   -1.499625   -0.028098
  C     -3.768181    1.641701    0.010433
  C     -2.832985    2.651061    0.006924
  C      2.228928    2.835679   -0.028158
  C      3.228183    1.912006   -0.037679
  C      3.437220   -3.149227   -0.053963
  C      2.502048   -4.158627   -0.050442
  C     -3.558669   -3.420017   -0.006390
  C     -2.559381   -4.343652   -0.015901
  H     -4.633051   -3.589818    0.000518
  H     -2.644718   -5.427990   -0.018393
  H     -0.016400   -5.285446   -0.035842
  H      2.681678   -5.229875   -0.055110
  H      4.519010   -3.247119   -0.061999
  H      4.366739   -0.548807   -0.052422
  H      2.314239    3.920088   -0.025633
  H      4.302487    2.081864   -0.044563
  H     -0.315062    3.778120   -0.007889
  H     -3.013895    3.722099    0.011827
```

(continues on next page)

(continued from previous page)

```
  H     -4.849852    1.740964    0.018679
  H     -4.697992   -0.958441    0.008649
end
```

In this case, all excitations up to 10 eV are considered from the very beginning. Configurations between 10 and 30 eV are included if their coupling to the configurations below 10 eV is strong enough (in total larger than `PThresh`). All configurations beyond 14 eV are neglected. Note furthermore that for very large systems, using a functional with the correct asymptotic behaviour is very important (due to the fixed amount of GGA exchange, CAM-B3LYP does **not** provide this property).

The ORCA output will summarize the important properties of your calculation which allows you to check your input:

```
-----------------------------------------------------------------------------

                       ORCA sTD-DFT CALCULATION

                       please cite in your paper
 orginal sTDA method:  S. Grimme, J. Chem. Phys. 138, 244104 (2013)
 range-separated sTDA: T. Risthaus, A. Hansen, S. Grimme, Phys. Chem. Chem. Phys.
                       16, 14408-14419 (2014)
 sTD-DFT approach:     C. Bannwarth, S. Grimme, Comp. Theor. Chem.
                       1040-1041, 45-53 (2014)

-----------------------------------------------------------------------------


spectral range up to (eV)      ...   10.000000
occ. MO cut-off (eV)           ...   -27.751992
virt. MO cut-off (eV)          ...   19.711835
perturbation threshold         ...   1.000e-04
CSF selection range up to (eV) ...   30.000000
MOs in sTD-DFT                 ...         150
occ. MOs in sTD-DFT            ...          55
virt. in sTD-DFT               ...          95
calculate triplets           ...   no

Transforming dipole length integrals   ... (read-ok)(trafo-ok)
Transforming dipole velocity integrals ... (read-ok)(trafo-ok)
Transforming ang. momentum integrals   ... (read-ok)(trafo-ok)

SCF atom population (using active MOs):

   4.196   5.171   5.171   4.196   3.616   3.996   3.991   3.991   3.996   3.616
   3.616   0.796   3.996   3.991   3.991   3.996   3.616   0.796   3.893   3.893
   4.040   4.040   3.893   3.893   4.040   4.040   0.971   0.971   0.956   0.957
   0.957   0.956   0.971   0.971   0.956   0.957   0.957   0.956

Number of electrons in sTDA:  110.000

ax(DF)   :  0.3800
s^k      :  2.0000
beta  (J):  1.8600
alpha (K):  0.9000
```

The spectroscopic data is also printed out after the calculation has finished:

```
-------------------------------------------------------------------------
         ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------
State   Energy  Wavelength   fosc         T2         TX         TY         TZ
        (cm-1)    (nm)                  (au**2)     (au)       (au)       (au)
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------------
   0   16964.7    589.5   0.000083715   0.00162   0.02872  -0.02828  -0.00022
   1   19477.4    513.4   0.001269789   0.02146  -0.09723  -0.10958   0.00035
   2   25753.4    388.3   1.245683315  15.92387   2.92682  -2.71234  -0.02807
   3   26309.7    380.1   1.402883697  17.55424   2.84893   3.07209  -0.01073
   4   30311.6    329.9   0.000000355   0.00000  -0.00146  -0.00131   0.00001
   5   32710.4    305.7   0.000000640   0.00001   0.00178  -0.00181   0.00000
   6   33285.3    300.4   0.349285082   3.45464   1.37427  -1.25134  -0.01315
   7   33622.4    297.4   0.097718276   0.95680   0.64724   0.73340  -0.00233
   8   34974.4    285.9   0.000000018   0.00000  -0.00024   0.00024   0.00024
   9   34994.6    285.8   0.000000073   0.00000  -0.00005  -0.00002   0.00082
  10   35723.1    279.9   0.000000022   0.00000  -0.00021  -0.00022  -0.00033
  11   35826.4    279.1   0.008160622   0.07499   0.00215  -0.00104   0.27383
  12   36282.4    275.6   0.000000000   0.00000  -0.00004   0.00000   0.00000
  13   37373.0    267.6   0.000000104   0.00000  -0.00067   0.00067  -0.00012
  14   37439.7    267.1   0.000000030   0.00000   0.00036   0.00037   0.00000
  15   38835.7    257.5   0.101992631   0.86460   0.63387   0.68029  -0.00240
  16   40111.9    249.3   0.000001866   0.00002  -0.00286   0.00267   0.00010
  17   40376.1    247.7   0.119778731   0.97663   0.72310  -0.67358  -0.00678
  18   41527.5    240.8   0.000000020   0.00000   0.00024   0.00031  -0.00000
  19   43202.8    231.5   0.000000006   0.00000   0.00015  -0.00014  -0.00008
  20   45265.3    220.9   0.000000008   0.00000   0.00013  -0.00014   0.00014
  21   46775.7    213.8   0.000000000   0.00000   0.00002   0.00003   0.00001
  22   49449.3    202.2   0.151078886   1.00582   0.73590  -0.68133  -0.00709
  23   51144.8    195.5   0.000000001   0.00000   0.00005  -0.00007   0.00003
  24   51371.9    194.7   0.000000069   0.00000  -0.00047  -0.00047  -0.00000
  25   53195.6    188.0   0.002581321   0.01598   0.00537  -0.00532  -0.12617
  26   53198.4    188.0   0.000000045   0.00000  -0.00031  -0.00038  -0.00020
  27   53341.2    187.5   0.065343188   0.40329   0.46944  -0.42767  -0.00298
  28   54136.6    184.7   0.006562015   0.03990   0.13280   0.14923  -0.00049
  29   54279.9    184.2   0.086565034   0.52502   0.49226   0.53170  -0.00183
  30   54285.4    184.2   0.078803195   0.47790  -0.46967  -0.50725   0.00179
  31   54351.4    184.0   0.000000001   0.00000   0.00005   0.00004  -0.00006
  32   54428.8    183.7   0.000000565   0.00000   0.00124   0.00137  -0.00011
  33   55214.4    181.1   0.130630885   0.77888   0.64617  -0.60109  -0.00629
  34   55671.1    179.6   0.000000219   0.00000   0.00083  -0.00077   0.00011
  35   56264.4    177.7   0.000000097   0.00000   0.00051   0.00055  -0.00000
  36   56499.9    177.0   0.013901825   0.08100  -0.19211  -0.20999   0.00072
  37   56694.8    176.4   0.000000005   0.00000  -0.00012  -0.00011   0.00000
  38   56885.1    175.8   0.000000018   0.00000  -0.00022  -0.00024  -0.00001
  39   57366.4    174.3   0.000144667   0.00083   0.01438   0.02497  -0.00003
  40   57498.6    173.9   0.000000213   0.00000  -0.00082   0.00074  -0.00002
  ...
```

5.6.9 Double-hybrid functionals and Doubles Correction

The program can compute a doubles correction to the CIS excitation energies. The theory is due to Head-Gordon and co-workers.[636] The basic idea is to compute a perturbative estimate (inspired by EOM-CCSD theory) to the CIS excited states that is compatible with the MP2 ground state energy. In many cases this is a significant improvement over CIS itself and comes at a reasonable cost since the correction is computed *a posteriori*. Of course, if the CIS prediction of the excited state is poor, the (D) correction – being perturbative in nature – cannot compensate for qualitatively wrong excited state wavefunctions.

In addition – and perhaps more importantly – the (D) correction is compatible with the philosophy of the double-hybrid functionals and should be used if excited states are to be computed with these functionals. The results are usually much better than those from TD-DFT since due to the large fraction HF exchange, the self-interaction error is much smaller than for other functionals and after the (D) correction the results do not suffer from the overestimation of transition energies that usually comes with increased amounts of HF exchange in TD-DFT calculations.

Since the calculations would require a fairly substantial integral transformation that would limit it to fairly small

molecules if no approximation are introduced we have decided to only implement a RI version of it. With this approximation systems with more than 1000 basis functions are readily within the reach of the implementation.

Since one always has a triad of computational steps: MP2-CIS solution-(D) correction, we have implemented several algorithms that may each become the method of choice under certain circumstances. The choice depends on the size of the system, the number of roots, the available main memory and the available disk space together with the I/O rate of the system. The formal cost of the (D) correction is $O(N^5)$ and its prefactor is higher than that of RI-MP2. In the best case scenario, the rate limiting step would be the calculation of the pair-contribution in the "U-term" which requires (for a closed-shell system) twice the effort of a RI-MP2 calculation *per state*.

The use of the (D)-correction is simple. Simply write:

```
! HF DEF2-SVP DEF2-SVP/C TightSCF
%cis dcorr n # n=1-4. The meaning of the four algorithms is
             # explained below.
   # algorithm 1 Is perhaps the best for small systems. May use a
   #             lot of disk space
   # algorithm 2 Stores less integrals
   # algorithm 3 Is good if the system is large and only a few
   #             states are calculated. Saves disk and main
   #             memory.
   # algorithm 4 Uses only transformed RI integrals. May be the
   #             fastest for large systems and a larger number
   #             of states
end
```

Table 5.7: Integral handling in various implementations of the (D) correction (i,j=occupied MOs, a,b=virtual MOs, Q=aux function; NumInt=numerical integration).

DCORR =	**1**	**2**	**3**	**4**
(ia|jb) integrals	Stored	Stored	Not stored	Not stored
(ij|ab) integrals	Stored	Not made	Not made	Not made
(ab|Q) integrals	Stored	Not made	Not made	Stored
(ij|Q) integrals	Stored	Stored	Stored	Stored
(ia|Q) integrals	Stored	Stored	Stored	Stored
Coulomb CIS	From (ia|jb)	From (ia|jb)	From (ia|Q)	From (ia|Q)
Exchange CIS	From (ij|ab)	RI-AO-direct	RI-AO-direct	From (ab|Q)
XC-CIS	Num. Int.	Num. Int.	Num. Int.	Num. Int.
V-term in (D)	From (ia|jb)	From (ia|jb)	From (ia|Q)	From (ia|Q)
U-term in (D)	From (ab|Q)	RI-AO-direct	RI-AO-direct	From (ab|Q)

Spin-component scaling versions of CIS(D) can be evoked in the %cis block by setting DOSCS TRUE and the four scaling parameters, as defined by Head-Gordon and co-workers [637], in the following order: same-spin indirect term (CTss), opposite-spin indirect term(CTos), same-spin direct term(CUss), and opposite-spin direct term(CUos). Note that this implementation only works for the version with the parameter $\lambda = 1$ as defined in Ref. [637]. The example below shows how to apply the SCS-CIS(D) version with $\lambda = 1$ whose usage has been advocated in Ref. [638]. The user is able to specify other scaling parameters.

```
%cis
dcorr     # n=1,2,3,4
doscs     true                    # set SCS-CIS(D) to true (default: false)
scspar    0.333, 1.2, 0.43, 1.24  #SCS-CIS(D) scaling parameters in this order
                                  CTss, CTos, CUss, CUos
end
```

Note the use of commas to separate the parameters. These parameters do not communicate with the SCS/SOS parameters set for ground-state SCS/SOS-MP2 in the %mp2 block.

Note

- CIS(D) is often a quite big improvement over CIS.
- In all three involved code sections (MP2, CIS, (D)) the storage format FLOAT is respected. It cuts down use of disk and main memory by a factor of two compared the default double precision version. The loss of accuracy should be negligible; however it is – as always in science – better to double check.
- The (ab|Q) list of integrals may be the largest for many systems and easily occupies several GB of disk space (hence algorithms 2 and 3). However, that disk-space is often well invested unless you run into I/O bottlenecks.
- The (ia|jb) and (ij|ab) lists of integrals is also quite large but is relatively efficiently handled. Nevertheless, I/O may be a problem.
- The cost of the (D) correction is O(N^5) and therefore comparable to RI-MP2. Since there are quite a few things more to be done for (D) compared to RI-MP2, expect the calculations to take longer. In the most elementary implementation the cost is about two times the time for RI-MP2 for each root.
- The (D) correction is compatible with the philosophy of the double-hybrid density functionals and should be used if these functionals are combined with TD-DFT. The program takes this as the default but will not enforce it. The (D) correction can be used both in a TD-DFT and TDA-DFT context.
- In our implementation it is only implemented together with the RI approximation and therefore you need to supply an appropriate ("/C") fitting basis.
- The program will automatically put the RI-MP2 module into operation together with the (D) correction. This will result in the necessary integrals becoming available to the CIS module.
- Making the exchange contribution to the CIS residual vector in an RI-AO direct fashion becomes quite expensive for a larger number of states. It may be a good choice if only one or two excited states are to be calculated for a larger system.
- Singlet-triplet excitations can be calculated by setting TRIPLETS TRUE in the %cis or %tddft blocks, respectively. The implementation has been tested for double hybrids in Ref. [639].
- For spin-adapted triplets (TRIPLETS TRUE), the only option available currently is DCORR 1.
- Spin-component and spin-opposite scaling techniques for double-hybrids within the TD- and TDA-DFT frameworks, as defined by Schwabe and Goerigk [640], can be evoked in the same way in the %tddft block as described for SCS-CIS(D) above. While user-defined parameters can be entered in such a way, a series of new functionals are available through normal keywords, which use the herein presented SCS/SOS-CIS(D) implementation. [266] See Sec. *Basic Usage* for a list of those functionals.
- Usage of time-dependent double-hybrids should be cited as follows: For TD or TDA with any double hybrid,[641] TD-B2GPLYP,[642] TDA-PBE0-DH or TDA-PBE0-2,[643] TD-PBE0-DH, TD-PBE0-2, or TDA-B2GP-PLYP [640], TD-ωB2PLYP or TD-ωB2GPPLYP [270], TDA-ωB2PLYP or TDA-ωB2GPPLYP [639], TD(A)-RSX-QIDH or TD(A)-RSX-0DH [639], TDA-PBE-QIDH [268], TD-PBE-QIDH [644], TD(A)-DSD-BLYP or TD(A)-DSD-PBEP86 or many other spin-component-scaled double-hybrid functionals with TD(A)-DFT from 2017 [640], TD(A) ωB88PP86 or TD(A) ωPBEPP86 or many other spin-component and opposite scaled double hybrids with TD(A)-DFT from 2021 [266].
- Note that SCS/SOS-CIS(D) is only automatically used when a TD(A)-DFT calculation is requested for the functionals from 2021 by Casanova-Páez and Goerigk. [266] In those instances, "doscs" has not to be set. SCS/SOS-CIS(D) is not automatically used for PWPB95, ωwB97X-2, or the DSD functionals.

5.6.10 Spin-orbit coupling

It also possible to include spin-orbit coupling between singlets and triplets calculated from TD-DFT by using quasi-degenerate perturbation theory (please refer to the relevant publication [601]), similarly to what is done in ROCIS. In order to do that, the flag DOSOC must be set to TRUE. The reduced matrix elements are printed and the new transition dipoles between all SOC coupled states are also printed after the regular ones. This option is currently still not compatible with double hybrids, but works for all other cases including CPCM. All the options regarding the SOC integrals can be altered in the %rel block, as usual.

```
%CIS DOSOC  TRUE END
```

Please have in mind that, as it is, you can only calculate the SOC between excited singlets and the spin-adapted triplets. There is no SOC starting from a UHF/UKS wavefunction. If you want more information printed such as the full SOC matrix or triplet-triplet couplings, please set a higher PRINTLEVEL.

SOC and ECPs

ORCA currently does not have SOC integrals for ECPs, and these are by default ignored in the SOC module. If you try to use ORCA together with ECPs, an abort message will be printed. If you absolutely need to use ECPs, for instance for embedded potentials, please use:

```
%TDDFT FORCEECP TRUE END
```

OBS.: Do not use ECPs in atoms where SOC might be important. In that case, always use all-electron basis functions or the results will not make sense.

5.6.11 Natural Transition Orbitals

Results of TD-DFT or CIS calculations can be tedious to interprete as many individual MO pairs may contribute to a given excited state. In order to facilitate the analysis while keeping the familiar picture of an excited state originating from essentially an electron being promoted from a donor orbital to an acceptor orbital, the concept of "natural transition orbitals" can be used.

The procedure is quite straightforward. For example, consider the following job on the pyridine molecule:

```
! PBE def2-SVP tightscf

%tddft
 nroots 5
 DoNTO  true      # flag to turn on generation of natural transition orbitals
 NTOStates 1,2,3  # States to consider for NTO analysis;
                  # if empty all will be done
 NTOThresh 1e-4   # threshold for printing occupation numbers
end

* xyz 0 1
  N      0.000000    0.000000    1.401146
  C      0.000000    1.146916    0.702130
  C      0.000000   -1.146916    0.702130
  C     -0.000000    1.205574   -0.702848
  C     -0.000000   -1.205574   -0.702848
  C      0.000000   -0.000000   -1.421344
  H     -0.000000    2.079900    1.297897
  H     -0.000000   -2.079900    1.297897
  H     -0.000000    2.179600   -1.219940
  H     -0.000000   -2.179600   -1.219940
  H      0.000000    0.000000   -2.525017
*
```

which results in:

```
-----------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE    1
-----------------------------------------

Making the (pseudo)densities                       ... done
Solving eigenvalue problem for the occupied space  ... done
Solving eigenvalue problem for the virtual space   ... done
Natural Transition Orbitals were saved in example10.s1.nto
Threshold for printing occupation numbers 0.000100

 E=   0.157831 au      4.295 eV    34639.9 cm**-1
    20a ->  21a  : n=  0.99945993
    19a ->  22a  : n=  0.00018772
    18a ->  23a  : n=  0.00018562


-----------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE    2
-----------------------------------------

Making the (pseudo)densities                       ... done
Solving eigenvalue problem for the occupied space  ... done
Solving eigenvalue problem for the virtual space   ... done
Natural Transition Orbitals were saved in example10.s2.nto
Threshold for printing occupation numbers 0.000100

 E=   0.158973 au      4.326 eV    34890.5 cm**-1
    20a ->  21a  : n=  0.99857289
    19a ->  22a  : n=  0.00054014
    18a ->  23a  : n=  0.00039643
    17a ->  24a  : n=  0.00029339


-----------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE    3
-----------------------------------------

Making the (pseudo)densities                       ... done
Solving eigenvalue problem for the occupied space  ... done
Solving eigenvalue problem for the virtual space   ... done
Natural Transition Orbitals were saved in example10.s3.nto
Threshold for printing occupation numbers 0.000100

 E=   0.200016 au      5.443 eV    43898.4 cm**-1
    20a ->  21a  : n=  0.63243468
    19a ->  22a  : n=  0.36314467
    18a ->  23a  : n=  0.00120185
    17a ->  24a  : n=  0.00103035
    16a ->  25a  : n=  0.00063379
    15a ->  26a  : n=  0.00051555
    14a ->  27a  : n=  0.00040257
    13a ->  28a  : n=  0.00020605
    12a ->  29a  : n=  0.00013070
    11a ->  30a  : n=  0.00010297


-----------------------------
TD-DFT/TDA-EXCITATION SPECTRA
-----------------------------

Center of mass = (  0.0000,  0.0000,  0.0036)
Generating CIS transition densities           ... done
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------------------
↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)     (nm)                 (au**2)    (au)    ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A  ->  1-1A    4.294795   34639.9   288.7   0.000000000   0.00000   0.00000  -
↪0.00000  -0.00000
  0-1A  ->  2-1A    4.325874   34890.5   286.6   0.003830922   0.03615   0.19012  ␣
↪0.00000   0.00000
  0-1A  ->  3-1A    5.442702   43898.4   227.8   0.019114326   0.14335   0.00000  -
↪0.37861   0.00000
  0-1A  ->  4-1A    6.579556   53067.7   188.4   0.017637721   0.10942   0.00000  ␣
↪0.00000   0.33078
  0-1A  ->  5-1A    6.739163   54355.0   184.0   0.007560799   0.04579  -0.00000  -
↪0.00000   0.21399
```

We see that there is a weakly allowed transition (STATE 2) that is essentially totally composed of a single NTO pair (20a→21a : n= 0.99857289), while the third excited state (S3) is strongly allowed and requires two NTO pairs for its description (20a→21a : n= 0.63243468 and 19a→22a : n= 0.36314467).

These orbitals are shown below. It is evident that the state 2 donor orbital (NTO20) is a nitrogen lone pair and the acceptor orbital is a $\pi*$ orbital of the ring. For the state 3 the two NTO donor orbitals are comprised of a nearly degenerate set of π orbitals (they would be degenerate in the parent benzene) and the acceptor orbitals are a pair of nearly degenerate $\pi*$ orbitals. It is evident from this example that by looking at the NTOs one can obtain a nicely pictorial view of the transition process, even if many orbital pairs contribute to a given excited state in the canonical basis.

Fig. 5.22: Natural transition orbitals for the pyridine molecule in state 2 and state 3.

Similar analysis can be performed in the case of ROCIS and DFT/ROCIS calculations as it will be described in section *Natural Transition Orbitals/ Natural Difference Orbitals*.

5.6.12 Computational Aspects

RI Approximation (AO-Basis)

If the SCF calculation used the RI approximation it will also be used in the TD-DFT calculation. The RI approximation saves a large amount of time while giving close to identical results (the errors will usually be <0.1 eV) and is generally recommended. If the functional is a hybrid functional the RI-approximation will only be applied to the Coulomb term while the exchange will be treated as before. In the SCF you can use this feature with the keyword (`! RIJONX`). It will then also be used in the TD-DFT calculation. Again, the RIJCOSX approximation can be used in TD-DFT and CIS calculations and leads to very large speedups at virtually no loss in accuracy.

Integral Handling

If the SCF calculation is carried out in an integral direct fashion this will also be done in the CIS/TD-DFT calculation. Thus, no bottlenecks arising from large integral transformations or large disk space requirement arise in the calculations. An exception is the MO based RI approximations described in the previous section.

Valence versus Rydberg States

For valence excited states the usual orbital basis sets are reasonable. Thus, with polarized double-zeta basis sets sensible results are obtained. Especially DFT calculations have the nice feature of not being overly basis set dependent.

If Rydberg states are desired, you should make sure that diffuse functions are present in your basis set. You could always use the augmented-specific basis, e.g. DEF2-TZVPD, ma-DEF2-TZVP, or aug-cc-pVTZ, or add some extra diffuse basis to your regular basis. These can be added to any "normal" basis set. For example, the following example provides a rather high quality basis for excited state calculations that is based on the Ahlrichs basis set:

```
%basis
  # augment the carbon basis set by diffuse functions
  addgto 6
    s 1
       1 0.01 1.0
    p 1
       1 0.01 1.0
    d 1
       1 0.07 1.0
    end
end
```

Tip

If you want to augment a given basis set it is sensible to run a preliminary SCF calculation and use `%output print[p_basis] 2 end`. This will provide you with a detailed listing of basis functions and their exponents. You can then add additional s, p and perhaps d-functions with the `AddGTO` command as in the example above. It is sensible to decrease the exponent of the diffuse functions by roughly a factor of 3 from the smallest exponent in the original basis.

Restrictions for Range-Separated Density Functionals

Several restrictions apply for range-separated (hybrid as well as double-hybrid) density functionals. They are currently only implemented to work with the AO-based algorithm within the RIJONX, RIJCOSX, and NORI integral schemes. Additionally, the asymptotic correction has been disabled. However, the nuclear gradient for the excited states is now available, including for the triplets. Please no that the IROOTMULT flag must be set to TRIPLET under %CIS or %TDDFT in order to obtain that.

Printing Extra Gradients Sequentially

If you want to print extra gradients for external applications or any other reason, you can use the keywords SGRADLIST and TGRADLIST, for singlets and triplets. This will print the gradients sequentially after the CIS/TDDFT run. If you put 0 on the singlet list, the ground state gradient will also be added, always at the end.

```
%TDDFT SGRADLIST  0, 1, 2
       TGRADLIST  2, 3
END
```

In order to save this gradients in a text file, please use:

```
%METHOD STORECISGRAD TRUE END
```

5.6.13 Restarting TD-DFT Calculations

TD-DFT amplitudes are stored by default in the GBW file. This also means that TD-DFT calculations can be restarted, if a suitable GBW file is provided. The restart is relatively flexible. It allows for the geometry, basis set or density functional to change between the source and target calculation. However, the source calculation must at least have the same number of roots than the target calculation or more, fewer roots in the source is not allowed.
Here is a simple example:

```
* gzmt 0 1
  O
  H  1 1.0
  H  1 1.0 2 104.0
*

%compound
 New_Step
   ! B3LYP def2-SVP def2/J VeryTightSCF PModel
   %Base "FirstJob"
   %cis nroots   10
        triplets true
        tda      true
        end
  Step_end

  New_Step
   ! B3LYP def2-SVP def2/J VeryTightSCF
   %Base "SecondJob"
   %cis nroots   10
        triplets true
        tda      true
        Restart "FirstJob.gbw"
        end
  Step_end

endrun;
```

Note

- This is supposed to work for all combinations of TDA or RPA, singlets and triplets and also for spin-flip TD-DFT calculations.
- We make several consistency checks but it will certainly be possible to create nonsensical inputs that will fail – please proceed with caution.

5.6.14 Use of TD-DFT for the Calculation of X-ray Absorption Spectra

In principle X-ray absorption spectra are "normal" absorption spectra that are just taken in a special high-energy wavelength range. Due to the high energy of the radiation employed (several thousand eV), core-electrons rather than valence electrons are excited. This has two consequences: a) the method becomes element specific because the core-level energies divide rather cleanly into regions that are specific for a given element. b) the wavelength of the radiation is so short that higher-order terms in the expansion of the light-matter interaction become important. Most noticeably, quadrupole intensity becomes important.

X-ray absorption spectra can be generally divided into three regions: a) the pre-edge that corresponds to transitions of core electrons into low lying virtual orbitals that lead to bound states. b) the rising edge that corresponds to excitations to high-lying states that are barely bound, and c) the extended X-ray absorption fine structure region (EXAFS) that corresponds to electrons being ejected from the absorber atom and scattered at neighbouring atoms.

With the simple TD-DFT calculations described here, one focuses the attention on the pre-edge region. Neither the rising edge nor the EXAFS region are reasonably described with standard electronic structure methods and no comparison should be attempted. In addition, these calculations are restricted to K-edges as the calculation of L-edges is much more laborious and requires a detailed treatment of the core hole spin orbit coupling.

It is clearly hopeless to try to calculate enough states to cover all transitions from the valence to the pre-edge region. Hence, instead one hand-selects the appropriate donor core orbitals and only allows excitations out of these orbitals into the entire virtual space. This approximation has been shown to be justified.[645] One should distinguish two situations: First, the core orbital in question may be well isolated and unambiguously defined. This is usually the case for metal 1s orbitals if there is only one metal of the given type in the molecule. Secondly, there may be several atoms of the same kind in the molecule and their core orbitals form the appropriate symmetry adapted linear combinations dictated by group theory. In this latter case special treatment is necessary: The sudden approximation dictates that the excitations occurs from a local core orbital. In previous versions of the program you had to manually localize the core holes. In the present version there is an automatic procedure that is described below.

A typical example is $TiCl_4$. If we want to calculate the titanium K-edge, the following input is appropriate:

```
! BP86 x2c x2c-SVPall x2c/J TightSCF

%tddft
 OrbWin[0] = 0,0,-1,-1
 NRoots 25
 DecomposeFosc true
 DoFullSemiclassical true
end

* int 0 1
 Ti 0 0 0  0 0 0
 Cl 1 2 3 2.15 0 0
 Cl 1 2 3 2.15 109.4712 0
 Cl 1 2 3 2.15 109.4712 120
 Cl 1 2 3 2.15 109.4712 240
*
```

Note

- The absolute transition energies from such calculations are off by a few hundred electron volts due to the shortcomings of DFT. The shift is constant and very systematic for a given element. Hence, calibration is possible and has been done for a number of edges already. Calibration depends on the basis set!
- Electric quadrupole contributions and magnetic dipole contributions have been invoked with `DecomposeFosc true` (check section *One Photon Spectroscopy* for more information), which is essential for metal edges. For ligand edges, the contributions are much smaller.
- `OrbWin` is used to select the single donor orbital (in this case the metal 1s). The LUMO (45) and last orbital in the set (174) are selected automatically if "-1" is given. This is different from previous program versions where the numbers had to be given manually.

The output contains standard TD-DFT output:

```
-----------------------------------------------------------------------------------
↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)    (nm)                 (au**2)    (au)    ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1A  ->  1-1A   4864.738909 39236769.5     0.3   0.000000000   0.00000  -0.
↪00000   0.00000   0.00000
  0-1A  ->  2-1A   4864.738936 39236769.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000  -0.00000
  0-1A  ->  3-1A   4865.587026 39243610.0     0.3   0.000378325   0.00000  -0.
↪00041   0.00173   0.00000
  0-1A  ->  4-1A   4865.587084 39243610.5     0.3   0.000378325   0.00000  -0.
↪00000  -0.00000   0.00178
  0-1A  ->  5-1A   4865.587090 39243610.6     0.3   0.000378325   0.00000   0.
↪00173   0.00041   0.00000
  0-1A  ->  6-1A   4868.943422 39270681.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-1A  ->  7-1A   4872.449970 39298963.4     0.3   0.000822318   0.00001   0.
↪00000  -0.00000  -0.00262
  0-1A  ->  8-1A   4872.450113 39298964.6     0.3   0.000822428   0.00001   0.
↪00168  -0.00202   0.00000
  0-1A  ->  9-1A   4872.450248 39298965.7     0.3   0.000822896   0.00001   0.
↪00202   0.00168  -0.00000
  0-1A  -> 10-1A   4873.185027 39304892.1     0.3   0.000000000   0.00000   0.
↪00000   0.00000  -0.00000
  0-1A  -> 11-1A   4873.752986 39309473.0     0.3   0.001152303   0.00001   0.
↪00035  -0.00309   0.00000
  0-1A  -> 12-1A   4873.753219 39309474.8     0.3   0.001152646   0.00001  -0.
↪00001   0.00000   0.00311
  0-1A  -> 13-1A   4873.753350 39309475.9     0.3   0.001152359   0.00001  -0.
↪00309  -0.00035  -0.00001
  0-1A  -> 14-1A   4873.877226 39310475.0     0.3   0.000000003   0.00000  -0.
↪00000   0.00000   0.00000
  0-1A  -> 15-1A   4873.877345 39310476.0     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-1A  -> 16-1A   4881.110098 39368812.1     0.3   0.000000000   0.00000  -0.
↪00000  -0.00000  -0.00000
  0-1A  -> 17-1A   4881.110123 39368812.3     0.3   0.000000000   0.00000  -0.
↪00000  -0.00000   0.00000
  0-1A  -> 18-1A   4883.621665 39389069.2     0.3   0.000014919   0.00000   0.
↪00021  -0.00029  -0.00000
```

(continues on next page)

(continued from previous page)

```
  0-1A  -> 19-1A   4883.621737 39389069.8     0.3   0.000014917   0.00000  -0.
↪00000   0.00000  -0.00035
  0-1A  -> 20-1A   4883.621842 39389070.7     0.3   0.000014922   0.00000   0.
↪00029   0.00021  -0.00000
  0-1A  -> 21-1A   4884.177852 39393555.2     0.3   0.000158653   0.00000  -0.
↪00069   0.00092   0.00000
  0-1A  -> 22-1A   4884.178076 39393557.0     0.3   0.000158639   0.00000  -0.
↪00092  -0.00069   0.00001
  0-1A  -> 23-1A   4884.178104 39393557.2     0.3   0.000158643   0.00000  -0.
↪00001  -0.00001  -0.00115
  0-1A  -> 24-1A   4885.935396 39407730.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-1A  -> 25-1A   4885.941881 39407783.0     0.3   0.000000000   0.00000  -0.
↪00000  -0.00000   0.00000
```

One may obtain origin independent transition moments formulations which can be combined with the multipole moments up to 2nd order to regenerate the electric dipole, electric quadrupole and magnetic dipole contributions in either length or the velocity representations. This requires in addition to the electric dipole (D), electric quadrupole (Q) and magnetic dipole (m) intensities the corresponding electric dipole - magnetic quadrupole (DM) and the electric dipole - electric octupole (DO) intensities.[646][647]. See also section *Excited States with Restricted Open-shell CIS - ROCIS*.

These spectra are requested by:

```
DecomposeFosc   true
```

Check section *One Photon Spectroscopy* for more information.

Resulting in:

```
-------------------------------------------------------------------------------
↪---------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM (Origin Independent, Length)
-------------------------------------------------------------------------------
↪---------------------------------------------
    Transition      Energy     Energy  Wavelength fosc(D2)  fosc(M2)  fosc(Q2)␣
↪fosc(D2+M2+Q2+DM+DO) D2/TOT    M2/TOT    Q2/TOT
                     (eV)      (cm-1)    (nm)       (au)    (au*1e6)  (au*1e6)
-------------------------------------------------------------------------------
↪---------------------------------------------
  0-1A  ->  1-1A   4864.738909 39236769.5     0.3     0.00000   0.00000   3.57414 ␣
↪ 0.00000357413611   0.00000   0.00000   1.00000
  0-1A  ->  2-1A   4864.738936 39236769.7     0.3     0.00000   0.00000   3.57419 ␣
↪ 0.00000357419113   0.00000   0.00000   1.00000
  0-1A  ->  3-1A   4865.587026 39243610.0     0.3     0.00038   0.00000   2.91269 ␣
↪ 0.00014515284191   2.60639   0.00000   0.02007
  0-1A  ->  4-1A   4865.587084 39243610.5     0.3     0.00038   0.00000   2.91280 ␣
↪ 0.00014514667816   2.60650   0.00000   0.02007
  0-1A  ->  5-1A   4865.587090 39243610.6     0.3     0.00038   0.00000   2.91281 ␣
↪ 0.00014515283005   2.60639   0.00000   0.02007
  0-1A  ->  6-1A   4868.943422 39270681.2     0.3     0.00000   0.00000   0.00000 ␣
↪ 0.00000000001034   0.12373   0.00000   0.00436
  0-1A  ->  7-1A   4872.449970 39298963.4     0.3     0.00082   0.00000  66.02903 ␣
↪ 0.00020716626448   3.96936   0.00000   0.31872
  0-1A  ->  8-1A   4872.450113 39298964.6     0.3     0.00082   0.00000  66.03525 ␣
↪ 0.00020716617979   3.96990   0.00000   0.31875
  0-1A  ->  9-1A   4872.450248 39298965.7     0.3     0.00082   0.00000  66.06697 ␣
↪ 0.00020743853075   3.96694   0.00000   0.31849
  0-1A  -> 10-1A   4873.185027 39304892.1     0.3     0.00000   0.00000   0.00000 ␣
↪ 0.00000000001255   1.59564   0.00000   0.11619
```

(continues on next page)

(continued from previous page)

```
  0-1A  -> 11-1A   4873.752986 39309473.0     0.3     0.00115   0.00000  65.17922 ␣
↪ 0.00112267473065   1.02639   0.00000   0.05806
  0-1A  -> 12-1A   4873.753219 39309474.8     0.3     0.00115   0.00000  65.20318 ␣
↪ 0.00112279513057   1.02659   0.00000   0.05807
  0-1A  -> 13-1A   4873.753350 39309475.9     0.3     0.00115   0.00000  65.18429 ␣
↪ 0.00112283712761   1.02629   0.00000   0.05805
  0-1A  -> 14-1A   4873.877226 39310475.0     0.3     0.00000   0.00000   0.95985 ␣
↪ 0.00000096238863   0.00274   0.00000   0.99737
  0-1A  -> 15-1A   4873.877345 39310476.0     0.3     0.00000   0.00000   0.95957 ␣
↪ 0.00000096000219   0.00031   0.00000   0.99955
  0-1A  -> 16-1A   4881.110098 39368812.1     0.3     0.00000   0.00000  59.20440 ␣
↪ 0.00005920440172   0.00000   0.00000   1.00000
  0-1A  -> 17-1A   4881.110123 39368812.3     0.3     0.00000   0.00000  59.20493 ␣
↪ 0.00005920493511   0.00000   0.00000   1.00000
  0-1A  -> 18-1A   4883.621665 39389069.2     0.3     0.00001   0.00000   1.60824 ␣
↪-0.00004102527947  -0.36365  -0.00000  -0.03920
  0-1A  -> 19-1A   4883.621737 39389069.8     0.3     0.00001   0.00000   1.60871 ␣
↪-0.00004102945913  -0.36356  -0.00000  -0.03921
  0-1A  -> 20-1A   4883.621842 39389070.7     0.3     0.00001   0.00000   1.60930 ␣
↪-0.00004103096489  -0.36367  -0.00000  -0.03922
  0-1A  -> 21-1A   4884.177852 39393555.2     0.3     0.00016   0.00000   7.33000 ␣
↪-0.00001710675356  -9.27427  -0.00000  -0.42849
  0-1A  -> 22-1A   4884.178076 39393557.0     0.3     0.00016   0.00000   7.32938 ␣
↪-0.00001711175162  -9.27075  -0.00000  -0.42832
  0-1A  -> 23-1A   4884.178104 39393557.2     0.3     0.00016   0.00000   7.32928 ␣
↪-0.00001712730312  -9.26257  -0.00000  -0.42793
  0-1A  -> 24-1A   4885.935396 39407730.7     0.3     0.00000   0.00000   0.00000 ␣
↪-0.00000000000329  -1.98584  -0.00000  -0.11175
  0-1A  -> 25-1A   4885.941881 39407783.0     0.3     0.00000   0.00017   0.00000 ␣
↪ 0.00000000016832   0.01099   1.00335   0.00021
```

The multipole moments up to 2nd order:

- Only approximate origin independence is achieved by using the length approximation for distances from the excited atom up to about 5 Angstrom.
- Can form negative intensities which can be partly cured by using larger basis sets.

Starting from ORCA version 6.0 the full semi-classical ligth-matter interaction[647][648][649] can be computed by including the keyword:

```
DoFullSemiclassical true
```

Resulting in:

```
---------------------------------------------------------------------------
                      ABSORPTION SPECTRUM VIA FULL SEMI-CLASSICAL FORMULATION
---------------------------------------------------------------------------
     Transition      Energy     Energy  Wavelength   fosc(FFMIO)
                      (eV)      (cm-1)     (nm)
---------------------------------------------------------------------------
  0-1A  ->  1-1A   4864.738909 39236769.5     0.3     0.00000225252009
  0-1A  ->  2-1A   4864.738936 39236769.7     0.3     0.00000225274848
  0-1A  ->  3-1A   4865.587026 39243610.0     0.3     0.00021278365996
  0-1A  ->  4-1A   4865.587084 39243610.5     0.3     0.00021278361154
  0-1A  ->  5-1A   4865.587090 39243610.6     0.3     0.00021278187385
  0-1A  ->  6-1A   4868.943422 39270681.2     0.3     0.00000000000237
  0-1A  ->  7-1A   4872.449970 39298963.4     0.3     0.00024200529569
  0-1A  ->  8-1A   4872.450113 39298964.6     0.3     0.00024205635341
  0-1A  ->  9-1A   4872.450248 39298965.7     0.3     0.00024227298357
  0-1A  -> 10-1A   4873.185027 39304892.1     0.3     0.00000000004292
  0-1A  -> 11-1A   4873.752986 39309473.0     0.3     0.00086416333614
```

(continues on next page)

(continued from previous page)

```
  0-1A  -> 12-1A   4873.753219 39309474.8     0.3     0.00086431970479
  0-1A  -> 13-1A   4873.753350 39309475.9     0.3     0.00086419143873
  0-1A  -> 14-1A   4873.877226 39310475.0     0.3     0.00000011039866
  0-1A  -> 15-1A   4873.877345 39310476.0     0.3     0.00000010856486
  0-1A  -> 16-1A   4881.110098 39368812.1     0.3     0.00000159577723
  0-1A  -> 17-1A   4881.110123 39368812.3     0.3     0.00000159558300
  0-1A  -> 18-1A   4883.621665 39389069.2     0.3     0.00000254791052
  0-1A  -> 19-1A   4883.621737 39389069.8     0.3     0.00000254735746
  0-1A  -> 20-1A   4883.621842 39389070.7     0.3     0.00000254890168
  0-1A  -> 21-1A   4884.177852 39393555.2     0.3     0.00008141388446
  0-1A  -> 22-1A   4884.178076 39393557.0     0.3     0.00008141175909
  0-1A  -> 23-1A   4884.178104 39393557.2     0.3     0.00008140437907
  0-1A  -> 24-1A   4885.935396 39407730.7     0.3     0.00000000004646
  0-1A  -> 25-1A   4885.941881 39407783.0     0.3     0.00000000003815
```

The full-semiclassical transition moments:

- Behave like the multipole expansion in the velocity representation.
- They are by definition origin independent they do not suffer from artificial negative values like the multipole moments beyond 1st order.

Note

- For information on plotting the obtained spectra check *orca_mapspc*

Now, let us turn to the Cl K-edge. Looking at the output of the first calculation, we have:

```
----------------
ORBITAL ENERGIES
----------------

  NO   OCC          E(Eh)            E(eV)
   0   2.0000    -178.943613     -4869.3033
   1   2.0000    -101.066445     -2750.1578
   2   2.0000    -101.066399     -2750.1565
   3   2.0000    -101.066377     -2750.1559
   4   2.0000    -101.066372     -2750.1558
   5   2.0000     -19.810459      -539.0700
   6   2.0000     -16.410626      -446.5558
   7   2.0000     -16.410625      -446.5558
   8   2.0000     -16.410625      -446.5558
   9   2.0000      -9.272000      -252.3039
  10   2.0000      -9.271962      -252.3029
  11   2.0000      -9.271945      -252.3024
  12   2.0000      -9.271940      -252.3023
  ...
```

And looking at the energy range or the orbital composition, we find that orbitals 1 through 4 are Cl 1s-orbitals. They all have the same energy since they are essentially non-interacting. Hence, we can localize them without invalidating the calculation. To this end, you can invoke the automatic localization for XAS which modifies the input to:

```
! BP86 x2c x2c-SVPall x2c/J TightSCF

%tddft
 XASLoc[0] = 1,4
 OrbWin[0] = 1,1,-1,-1
 NRoots    25
 DecomposeFosc  true
 DoFullSemiclassical true
```

(continues on next page)

(continued from previous page)

```
end

* int 0 1
 Ti 0 0 0  0 0 0
 Cl 1 2 3 2.15 0 0
 Cl 1 2 3 2.15 109.4712 0
 Cl 1 2 3 2.15 109.4712 120
 Cl 1 2 3 2.15 109.4712 240
*
```

Note

- This localizes the orbitals 1 through 4 of operator 0 (the closed-shell) and then allows excitations (arbitrarily) from core hole 1 only. You could choose any of the three other localized 1s orbitals instead without changing the result. You could even do all four core holes simultaneously (they produce identical spectra) in which case you have the entire ligand K-edge intensity and not just the one normalized to a single chlorine (this would be achieved with `OrbWin[0] = 1,4,-1,-1`).
- If you have a spin unrestricted calculation, you need to give the same `XASLoc` and `OrbWin` information for the spin-down orbitals as well.

The obtained Cl K-edge absorption is:

```
-----------------------------------------------------------------------------------
-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX      
   DY        DZ
                      (eV)      (cm-1)     (nm)                 (au**2)    (au)     
  (au)       (au)
-----------------------------------------------------------------------------------
-----------------
  0-1A  ->  1-1A   2745.600263 22144761.9     0.5   0.000527911   0.00001   0.
00000   0.00000   0.00280
  0-1A  ->  2-1A   2745.600291 22144762.1     0.5   0.000527960   0.00001  -0.
00000  -0.00280   0.00000
  0-1A  ->  3-1A   2746.447743 22151597.3     0.5   0.000444357   0.00001   0.
00001  -0.00257  -0.00000
  0-1A  ->  4-1A   2746.447798 22151597.7     0.5   0.000444395   0.00001   0.
00000   0.00000  -0.00257
  0-1A  ->  5-1A   2746.458249 22151682.0     0.5   0.001391214   0.00002  -0.
00455  -0.00000  -0.00000
  0-1A  ->  6-1A   2749.820836 22178803.1     0.5   0.000108586   0.00000  -0.
00127   0.00000  -0.00000
  0-1A  ->  7-1A   2753.309672 22206942.5     0.5   0.000642499   0.00001  -0.
00000  -0.00000  -0.00309
  0-1A  ->  8-1A   2753.309873 22206944.1     0.5   0.000643000   0.00001  -0.
00004  -0.00309   0.00000
  0-1A  ->  9-1A   2753.320060 22207026.2     0.5   0.001717215   0.00003   0.
00505  -0.00002  -0.00000
  0-1A  -> 10-1A   2753.907046 22211760.6     0.5   0.001737693   0.00003  -0.
00507  -0.00000   0.00000
  0-1A  -> 11-1A   2754.608481 22217418.1     0.5   0.000027939   0.00000   0.
00000   0.00064  -0.00000
  0-1A  -> 12-1A   2754.608709 22217419.9     0.5   0.000027879   0.00000  -0.
00000   0.00000   0.00064
  0-1A  -> 13-1A   2754.709845 22218235.6     0.5   0.000149389   0.00000   0.
```

(continues on next page)

(continued from previous page)

```
↪00149  -0.00001   0.00000
  0-1A  -> 14-1A   2754.736803 22218453.0     0.5   0.000681956   0.00001  -0.
↪00001  -0.00318   0.00000
  0-1A  -> 15-1A   2754.736928 22218454.0     0.5   0.000682692   0.00001  -0.
↪00000   0.00000   0.00318
  0-1A  -> 16-1A   2761.966240 22276762.4     0.4   0.000069982   0.00000   0.
↪00000  -0.00000   0.00102
  0-1A  -> 17-1A   2761.966265 22276762.6     0.4   0.000069999   0.00000   0.
↪00000   0.00102   0.00000
  0-1A  -> 18-1A   2764.478344 22297023.9     0.4   0.000309542   0.00000   0.
↪00000  -0.00214  -0.00001
  0-1A  -> 19-1A   2764.478355 22297024.0     0.4   0.000309644   0.00000   0.
↪00000  -0.00001   0.00214
  0-1A  -> 20-1A   2764.662301 22298507.6     0.4   0.000422965   0.00001  -0.
↪00250  -0.00000   0.00000
  0-1A  -> 21-1A   2765.033563 22301502.0     0.4   0.000115414   0.00000   0.
↪00000   0.00131  -0.00000
  0-1A  -> 22-1A   2765.033736 22301503.4     0.4   0.000115508   0.00000   0.
↪00000   0.00000   0.00131
  0-1A  -> 23-1A   2765.243962 22303199.0     0.4   0.000222110   0.00000  -0.
↪00181   0.00000   0.00000
  0-1A  -> 24-1A   2766.796424 22315720.5     0.4   0.000000384   0.00000   0.
↪00000  -0.00000   0.00008
  0-1A  -> 25-1A   2766.796522 22315721.3     0.4   0.000002667   0.00000   0.
↪00000   0.00020   0.00000
```

Quite nice results have been obtained for a number of systems in this way.[650]

5.6.15 Transient spectra

If one wants to compute transient spectra, or transition dipoles starting from a given excited state, the option DOTRANS must be set to TRUE and an IROOT should be given for the initial state (the default is 1). If `DOTRANS ALL` is requested instead, the transition dipoles between all states are computed. The transient transition dipoles will then be printed after the normal spectra. This option is currently only available for CIS/TDA and is done usng the expectation value formalism, as the other transition dipole moments in ORCA.

```
%cis
  DOTRANS  TRUE
           #or
  DOTRANS  ALL
end
```

5.6.16 Excited State Geometry Optimization

For RPA, CIS, TDA and TD-DFT the program can calculate analytic gradients. With the help of the `IRoot` keyword, a given state can be selected for geometry optimization. Note however, that if two states cross during the optimization it may fail to converge or fail to converge to the desired excited state (see section *Root Following Scheme for Difficult Cases* below)!

Note

- If you want to follow a triplet state instead of the singlet, please include `IROOTMULT TRIPLET`.

```
!def2-SVP Opt Freq
```

(continues on next page)

(continued from previous page)

```
%cis
 NRoots    1
 IRoot     1
end

* int 0 1
    C  0 0 0 0.00 0.0     0.00
    O  1 0 0 1.20 0.0     0.00
    H  1 2 0 1.08 120     0.00
    H  1 2 3 1.08 120   180.00
*
```

Note that this example converges to a saddle point as can be verified through the numerical frequency calculation:

```
-----------------------
VIBRATIONAL FREQUENCIES
-----------------------

Scaling factor for frequencies =  1.000000000  (already applied!)

     0:       0.00 cm**-1
     1:       0.00 cm**-1
     2:       0.00 cm**-1
     3:       0.00 cm**-1
     4:       0.00 cm**-1
     5:       0.00 cm**-1
     6:    -365.27 cm**-1  ***imaginary mode***
     7:     949.68 cm**-1
     8:    1396.45 cm**-1
     9:    1688.28 cm**-1
    10:    3200.58 cm**-1
    11:    3313.95 cm**-1
```

The excited state relaxed density matrixis available from such gradient runs (`MyJob.cisp` when using the `KeepDens` keyword) and can be used for various types of analysis.

Root Following Scheme for Difficult Cases

In case there is a root flipping after a step during the geometry optimization, it might be impossible to converge an excited state geometry using the regular methods. To help in those cases, the flag FOLLOWIROOT might be set to TRUE. Then, excited state wavefunction will be analyzed and compared with the reference one (more below), and the IROOT will be automatically adjusted to keep homing the target state.

One example of such a calculation is:

```
! wB97X OPT

%tddft
 NRoots        5
 IRoot         3
 FOLLOWIROOT   TRUE
end

* xyz 0 1
 N 0.0  0.0  0.0
 H 0.0  0.0  1.0
 H 0.0 -0.9  0.5
 H 0.0  0.9  0.5
*
```

This will ask for an optimization of the third excited state of ammonia. At some point, there is a state crossing and what was state 3 now becomes state 2. The algorithm will recognize this and automatically change the `IROOT` flag, to keep following the same state:

```
The IROOT now is:                                ... 2
```

`FOLLOWIROOT` also works with spin-adapted triplets and spin-flip states.

In cases where you want to keep the comparison only with the density from **the very first** computed excited state, e.g. the one you get on the first cycle of a geometry optimization, you can use FIRKEEPFIRSTREF, as in:

```
%tddft
  NRoots           5
  IRoot            3
  FOLLOWIROOT      TRUE
  FIRKEEPFIRSTREF  TRUE # default false
end
```

Criteria to Follow IROOTs - starting from ORCA6

Starting from ORCA6 we have a much more robust algorithm to follow these excited states, inspired by some of the recent literature [651] [652]. The algorithm now works as follows, after each excited state calculation using CIS/TDDFT:

1. Given a reference state, take all states within an energy difference of up to 1 eV to it. We don't want to check states that are too far apart in energy. Controlled by `%TDDFT FIRENTHRESH 1.0 END`, number in eV.
2. Now take all states with a difference of $\hat{S}^2$ not larger than 0.5. We don't want to compare singlets to triplets. Controlled by `%TDDFT FIRS2THRESH 0.5 END`.
3. Calculate the overlap between the transition densities of all states with the reference - this is the core part.
4. In case there is ambiguity - that is if two states have overlaps differing by only 0.05 - take the one with the closer transition dipole angle. Controlled by `%TDDFT FIRSTHRESH 0.05 END`.
5. Update the IROOT to the state that went best on all these tests.

> **Note**
>
> These `FIR` keywords are specific to **F**ollow **IR**oot.

Now by default we might also update the reference state from time to time in case the separation of states is very clear. The way it work is:

1. If the best overlap is larger than `FIRMINOVERLAP`, which is 0.5 by default and `FIRDYNOVERLAP` is `FALSE`, we will assume that the overlap is good enough and we will always update the reference. However, the default is `FIRDYNOVERLAP TRUE`, which means also have a second check for robustness.
2. If `FIRDYNOVERLAP TRUE` and the best overlap is larger than 0.5 (or `FIRMINOVERLAP`), we will check for the `ratio` between the best and the second best states. If this `ratio` is between 0.3 and 0.6 (controlled by `%TDDFT FIRDYNOVERRATIO 0.3,0.6 END`), it means that there is a clear separation between the best and the second best and the reference can be updated safely. If the ratio is too close to 1, both states are too similar and it would be dangerous to update the reference state. If it is too close to zero, they are easy to distinguish and we don't need to update the reference yet [651].

> **Important**
>
> It is important to stress that this will not necessarily solve all problems (root flipping can be particularly bad if the system is highly symmetric), for the excited states may change too much during the optimization. If that happens, it is advisable to restart the calculation after some steps and check which IROOT you still want. This

can also be used when calculating numerical gradients and Hessians, in case you suspect of root flipping after the displacements.

Important

This algorithm is completely general and should work for any excited state method, as long as there are transition densities. We will include more methods in the future when possible.

Geometry Optimization of SOC States

If you want to compute geometries for the SOC states, just choose `SOCGRAD TRUE` and a given `IROOT`. The weigthed "unrelaxed" gradient will then be calculated after selecting the CIS/TD-DFT states with contribution larger than 0.01%. Each gradient will be calculated separately and, after that, the final SOC gradient will be computed as a weighted sum. Setting `IROOT 0` in this case corresponds to ask for the SOC ground state, which is NOT necessarily equal to the ground state from HF/DFT.

5.6.17 Potential Energy Surface Scans

ORCA allows the combination the scan feature with CIS or TD-DFT. This can be used to map out the excited state potential energy surfaces as a function of one- two- or three parameters. The output of the "trajectory" run automatically contains the excited state energies in addition to the ground state energy. For example consider the following simple job.

```
! def2-TZVPD

%method
 scanguess pmodel  # this assignment forces a PModel guess at each step
                   # which is often better if diffuse functions are present
end

%cis
NRoots 7
end

%paras
 rCO = 0.85,1.45,21;
end

* xyz 0 1
  O  0  0  0
  C  0  0  {rCO}
*
```

The output file from this job contains the *total* energies (i.e. the ground state energy plus the excitation energy) for each excited state as a function of C-O bondlength as shown below. Howerver, the assignment of the individual states will change with geometry due to curve crossings. Thus, the state-to-state correlation must be worked out "by hand". These calculations are nevertheless very helpful in obtaining at least a rough idea about excited state energy surfaces.

Fig. 5.23: Result of a potential energy surface scan for the excited states of the CO molecule using the `orca_cis` module.

Potential Energy Surface Scans along Normal Coordinates

The ground and excited state potential energy surfaces can also be mapped as a function of normal coordinates. The normal mode trajectory run is invoked by the keyword `!MTR`. In addition several parameters have to be specified in the block `%mtr`. The following example illustrates the use:

First you run a frequency job:

```
#
! BP86 def2-SV(P) def2/J TightSCF AnFreq

* xyz 0 1
  C       0.000001     -0.000000     -0.671602
  C       0.000000      0.000000      0.671602
  H      -0.000000     -0.940772     -1.252732
  H      -0.000000     -0.940772      1.252732
  H      -0.000000      0.940772     -1.252732
  H      -0.000000      0.940772      1.252732
*
```

and then:

```
! BP86 def2-SV(P) def2/J TightSCF MTR

%tddft
  NRoots 3
  triplets false
 end

%mtr
   HessName "ethene.hess"
   modetype normal
   MList 9,13
   RSteps   4,5
   LSteps   4,5
   ddnc     1.0, 0.5
end

* xyz 0 1
  C       0.000001     -0.000000     -0.671602
  C       0.000000      0.000000      0.671602
  H      -0.000000     -0.940772     -1.252732
  H      -0.000000     -0.940772      1.252732
  H      -0.000000      0.940772     -1.252732
  H      -0.000000      0.940772      1.252732
*
```

The `HessName` parameter specifies the name of the file which contains nuclear Hessian matrix calculated in the frequency run. The Hessian matrix is used to construct normal mode trajectories. The keyword `MList` provides the list of the normal modes to be scanned. The parameters `RSteps` and `LSteps` specify the number of steps in positive and negative direction along each mode in the list. In general, for a given set of parameters

```
mlist m1,m2,...mn
rsteps rm1,rm2,...rmn
lsteps lm1,lm2,...lmn
```

the total number of the displaced geometries for which single point calculations will be performed is equal to $\prod_{m_i} (r_{m_i} + l_{m_i} + 1)$. Thus, in the present case this number is equal to $(4 + 4 + 1)\,(5 + 5 + 1) = 99$.

The `ddnc` parameter specifies increments δq_α for respective normal modes in the list in terms of dimensionless normal coordinates (DNC's). The trajectories are constructed so that corresponding normal coordinates are varied in the range from $-l_\alpha \delta q_\alpha$ to $r_\alpha \delta q_\alpha$. The measure of normal mode displacements in terms DNC's is appropriate choice

since in spectroscopical applications the potential energy function U is usually expressed in terms of the DNC's. In particular, in the harmonic approximation $U(q_\alpha)$ has a very simple form around equilibrium geometry:

$$U = U_0 + \sum_{\alpha}^{3N-6} \frac{\hbar\omega_\alpha}{2} q_\alpha^2 \tag{5.79}$$

where $\omega_\alpha$is the vibrational frequency of the α-th mode.

Dimensionless normal coordinate q_α can be related to the vector of atomic Cartesian displacements $\delta\mathbf{X}$ as follows:

$$q_\alpha = \left(\frac{\omega_\alpha}{\hbar}\right)^{\frac{1}{2}} \sum_{k=1}^{3N} L_{k\alpha} \delta X_k \sqrt{M_k} \tag{5.80}$$

where $\{L_{k\alpha}\}$ is the orthogonal matrix obtained upon numerical diagonalization of the mass-weighted Hessian matrix, and $\mathbf{M}$ is the vector of atomic masses. Accordingly, the atomic Cartesian displacements corresponding to a given dimensionless normal coordinate q_α are given by:

$$\delta X_k = \left(\frac{\hbar}{\omega_\alpha}\right)^{\frac{1}{2}} L_{k\alpha} q_\alpha \left(M_k\right)^{-\frac{1}{2}} \tag{5.81}$$

Alternatively, it is possible to specify in the input the Cartesian increment for each normal mode. In such a case, instead of the `ddnc` parameter one should use the `dxyz` keyword followed by the values of Cartesian displacements, for example:

```
%mtr
   HessName "ethene.hess"
   modetype normal
   MList 9,13
   RSteps   4,5
   LSteps   4,5
   dxyz     0.01, 0.02 # increments in the Cartesian basis
                       # are given in angstrom units
end
```

For a given Cartesian increment $d_{X,\alpha}$ along the α–th normal mode the atomic displacements are calculated as follows:

$$\delta X_k = \frac{d_{X,\alpha}}{\|\mathbf{T}_\alpha\|} L_{k\alpha} \left(M_k\right)^{-\frac{1}{2}} \tag{5.82}$$

The vector $\mathbf{T}_\alpha$ in the Cartesian basis has components $T_{i\alpha} = L_{k\alpha} \left(M_k\right)^{-\frac{1}{2}}$ and length (norm) $\|\mathbf{T}_k\|$.

The increment length can also be selected on the basis of an estimate for the expected change in the total energy ΔE due to the displacement according to eq.(3.92). The value of ΔE can be specified via the `EnStep` parameter:

```
%mtr
   HessName "ethene.hess"
   modetype normal
   MList 9,13
   RSteps   4,5
   LSteps   4,5
   EnStep   0.001, 0.001 # the values are given in Eh
end
```

All quantum chemical methods have to tolerate a certain amount of numerical noise that results from finite convergence tolerances or other cutoffs that are introduced into the theoretical procedures. Hence, it is reasonable to choose ΔE such that it is above the characteristic numerical noise level for the given method of calculation.

At the beginning of the program run the following trajectory files which can be visualized in gOpenMol will be created:

- `BaseName.m9.xyz` and `BaseName.m13.xyz` contain trajectories along normal modes 9 and 13, respectively.

- `BaseName.m13s1.m9.xyz - BaseName.m13s5.m9.xyz` contain trajectories along normal mode 9 for different fixed displacements along mode 13, so that the file `BaseName.m13sn.m9.xyz` corresponds to the n-th step in the positive direction along mode 13.
- `BaseName.m13s-1.m9.xyz - BaseName.m13s-5.m9.xyz` contain trajectories along normal mode 9 for different fixed displacements along mode 13, so that the file `BaseName.m13s-n.m9.xyz` corresponds to the n-th step in the negative direction along mode 13.
- `BaseName.m9s1.m13.xyz - BaseName.m9s4.m13.xyz` contain trajectories along normal mode 13 for different fixed displacements along mode 9, so that the file `BaseName.m9sn.m13.xyz` corresponds to the n-th step in the positive direction along mode 9.
- `BaseName.m9s-1.m13.xyz - BaseName.m9s-4.m13.xyz` contain trajectories along normal mode 13 for different fixed displacements along mode 9, so that the file `BaseName.m9s-n.m13.xyz` corresponds to the n-th step in the negative direction along mode 9.

The results of energy single point calculations along the trajectories will be collected in files `BaseName.mtr.escf.S.dat` (for the SCF total energies) and files `BaseName.mtr.ecis.S.dat` (for the CIS/TDDFT total energies), where "S" in the suffix of `*.S.dat` filenames provides specification of the corresponding trajectory in the same way as it was done for the case of trajectory files `*.xyz` (e.g. `S="m9s-1.m13"`). Likewise, the calculated total energies along the trajectories will be collected in files `BaseName.mtr.emp2.S.dat` in the case of MP2 calculations, `BaseName.mtr.emdci.S.dat` (MDCI), `BaseName.mtr.ecasscf.S.dat` (CASSCF), `BaseName.mtr.emrci.S.dat` (MRCI).

Note

In principle normal coordinate trajectories can be performed for an arbitrary number normal modes. This implies that in general trajectories will contain geometries which involve simultataneous displacement along several (>2) modes. However, trajectory files `*.xyz` and corresponding `*.dat` files will be generated only for the structures which are simultaneously displaced along not more than 2 normal coordinates.

Fig. 5.24: Result of a potential energy surface scan along C-C stretching normal coordinate (mode 13 in the present example) for the excited states of the ethene molecule using the `orca_cis` module.

Normal Mode Scan Calculations Between Different Structures

This type of job allows to map PES between two different structures as a function of normal coordinates. The H_2O molecule represent a trivial case which has formally 2 equivalent equilibrium structures which differ by angle H_1—O—H_2 (103.5° and 256.5°, respectively, as follows from the BP86/SV(P) calculations). In such a case the input for the nomal mode trajectory run would require the calculation of geometry difference between both structures in terms of the dimensionless normal coordinates. This can be done in `orca_vib` run as follows :

```
> orca_vib water.hess ddnc geom.xyz
```

The second parameter ddnc in the command line invokes the calculation of geometry difference in terms of the DNC's. Both structures are specified in the file `geom.xyz` which has a strict format:

```
2  3
0
        0.000000      0.000000      0.000000
        0.000000      0.607566      0.770693
        0.000000      0.607566     -0.770693
1
        0.000000      0.000000      0.000000
        0.000000     -0.607566      0.770693
        0.000000     -0.607566     -0.770693
```

The first line of the input specifies the number of the structures and total number of atoms (2 and 3, respectively). Specification of each structure in sequence starts with a new line containing the number of the structure. The number 0 in the second line is used to denote the reference structure. Note that atomic coordinates should be given in units of Å and in the same order as in the ORCA input for the frequency run from which the file `water.hess` was calculated.

At the end of the `orca_vib` run the file `geom.ddnc` is generated. It contains the geometry difference in terms of the dimensionless normal coordinates between the structures with nonzero numbers and the reference one in `geom.xyz`:

```
1
1 9
    0       0.000000
    1       0.000000
    2       0.000000
    3       0.000000
    4       0.000000
    5       0.000000
    6       9.091932
    7      -9.723073
    8       0.000000
```

The output file indicates that the structural difference occurs along 2 normal coordinates: 6 (bending mode) and 7 (totally symmetric O—H stretching mode). On the basis of the calculated displacement pattern the following input for the normal mode trajectory run between two structures can be designed:

```
! RKS BP86 SV(P) def2/J  RI TightScf MTR

%mtr
   HessName "water.hess"
   modetype normal
   mlist 6,7
   rsteps   10,0
   lsteps   0, 10
   ddnc     0.9091932, 0.9723073
end

* xyz 0 1
```

(continues on next page)

(continued from previous page)

```
O        0.000000      0.000000      0.000000
H        0.000000      0.607566      0.770693
H        0.000000      0.607566     -0.770693
*
```

Here the parameters `RSteps`, `LSteps` and `ddnc` are chosen in such a way that in the scan along modes 6 and 7 the corresponding dimensionless normal coordinates will be varied in the range 0 − 9.091932 and -9.723073 − 0, respectively, in accordance with the projection pattern indicated in the file `geom.ddnc`. Note that normal modes are only defined up to an arbitrary choice of sign. Consequently, the absolute sign of the dimensionless displacements is ambiguous and in principle can vary in different `orca_vib` runs. It is important that the normal mode scan between different structures exemplified above is performed using the same sign of normal modes as in the calculation of normal mode displacements. This condition is fulfilled if the same normal modes are used in `orca_vib` run and trajectory calculation. Thus, since in `orca_vib` calculation normal modes are stored in `.hess` file it is necessary to use the same Hessian file in the trajectory calculation.

5.6.18 Non-adiabatic coupling matrix elements

The CIS module can compute the non-adiabatic coupling matrix elements (NACME) between ground and an excited state given by an IROOT, $\langle \Psi_{GS} | \frac{\partial}{\partial R_x} | \Psi_{IROOT} \rangle$ [653]. These can also include LR-CPCM effects if !CPCM(solvent) is chosen in the main input, ZORA effects and will make use of RIJ and COSX, if they are chosen for the SCF. The usage is simple, e.g.:

```
!PBE0 DEF2-SVP TIGHTSCF
%TDDFT  NROOTS  5
        IROOT   2
        NACME   TRUE
END
* xyz 0 1
O        0.000000000      0.000000000      0.611403292
C        0.000000000      0.000000000     -0.613232096
H        0.931880792      0.000000000     -1.200880848
H       -0.931880792      0.000000000     -1.200880848
*
```

By choosing NACME TRUE under %TDDFT, a regular gradient calculation will be done, and the NACMEs will be computed together with it. After the usual gradient output, the NACMEs will be printed as:

```
---------------------------------
CARTESIAN NON-ADIABATIC COUPLINGS
       <GS|d/dx|ES>
---------------------------------

1   O   :   -0.161958900   -0.000000135    0.000001029
2   C   :   -0.088213027   -0.000000024    0.000000167
3   H   :    0.226241398    0.000000001   -0.109102154
4   H   :    0.226241406   -0.000000000    0.109102161

Difference to translation invariance:
:    0.2023108777   -0.0000001585    0.0000012017

Norm of the NACs                   ...    0.4002363416
RMS NACs                           ...    0.1155382798
MAX NAC                            ...    0.2262414063
```

The NACMEs are given under the Cartesian basis; this may differ from some other programs where NACMEs are given under the normal mode basis. To obtain NACMEs under the normal mode basis, one can perform an internal conversion (IC) rate calculation (see *Internal Conversion Rates (unpublished)*) and obtain them from the output file.

NACMEs with built-in electron-translation factor

As you can see, the calculation above does not have full translation invariance! That is a feature of NACs calculated from CI wavefunctions, due to the Born-Oppenheimer approximation. It can be somehow fixed by including the so-called "electron-translation factors" (ETFs) [654], and those are added with ETF TRUE under %TDDFT. By now using the input:

```
!PBE0 DEF2-SVP
%TDDFT  NROOTS  5
     IROOT   2
     NACME   TRUE
     ETF     TRUE
END
* xyz 0 1
O        0.000000000      0.000000000      0.611403292
C        0.000000000      0.000000000     -0.613232096
H        0.931880792      0.000000000     -1.200880848
H       -0.931880792      0.000000000     -1.200880848
*
```

one gets the following output:

```
---------------------------------
CARTESIAN NON-ADIABATIC COUPLINGS
           <GS|d/dx|ES>
        with built-in ETFs
---------------------------------

1   O   :   -0.071334028   -0.000001941    0.000003727
2   C   :   -0.362514525   -0.000000130   -0.000000776
3   H   :    0.217014763    0.000000003   -0.128968922
4   H   :    0.217014813   -0.000000002    0.128968939

Difference to translation invariance:
:    0.0001810232   -0.0000020693    0.0000029689

Norm of the NACs                   ...    0.5137724505
RMS NACs                           ...    0.1483133313
MAX NAC                            ...    0.3625145251
```

where the residual translation variance is due to the DFT and COSX grids only.

> **Warning**
>
> These are the recommended NACs to be used with any kind of dynamics or conical intersection optimization, otherwise moving the center of mass of you system would already change the couplings!

Numerical non-adiabatic coupling matrix elements

The numerical non-adiabatic coupling matrix elements between ground and excited states from CIS/TD-DFT can be calculated in a numerical fashion, by setting the NumNACME flag on the main input line:

```
! NumNACME
```

ORCA will then calculate both the NACMEs and the numerical gradient for a given IROOT at the same cost. Please be careful with the SCF options and GRID sizes since there are displacements involved, for more information check *Numerical Gradients*. All options regarding step size and so on can be changed from %NUMGRAD.

These are current implemented in both RHF/RKS and UHF/UKS, but only for CIS/TDA and RPA/TD-DFT, no multireference methods yet. For the latter case, the overlap of the $|X - Y\rangle$ vector is used [655].

5.6.19 Keyword List

Table 5.8: `%tddft` or `%cis` block input keywords.

Keyword	Option / Value	Description
`NRoots`	`3`	The number of desired roots.
`IRoot`	`1`	The root to be optimized.
`IRootMult`	`singlet` `triplet`	Multiplicity of the root to be optimized.
`MaxDim`	`5`	Davidson expansion space = MaxDim * NRoots.
`MaxIter`	`35`	Maximum CI Iterations.
`NGuessMat`	`512`	The dimension of the guess matrix.
`MaxCore`	`4096`	The maximum memory to be used on this calculation.
`ETol`	`1e-6`	Energy convergence tolerance.
`RTol`	`1e-6`	Residual Convergence tolerance.
`TDA`	`false`	Switch off for full TDDFT.
`LRCPCM`	`true`	Use LRCPCM.
`CPCMEQ`	`false`	Which epsilon is used to compute the charges.
`DoNTO`	`true`	Generate Natural Transition Orbitals. (For double hybrids, the NTOs do not contain PT2 contributions)
`NTOStates`	`1,2,3`	States to consider for NTO analysis. If empty, all will be done.
`NTOThresh`	`1e-4`	Threshold for printing occupation numbers.
`SaveUnrNatOrb`	`true`	Saves natural orbitals (not NTO) from unrelaxed densities for the IROOT chosen (including IROOTLISTs).
`DoSoc`	`false`	Include spin-orbit coupling.
`SocGrad`	`false`	Set true to compute the SOC gradient for a given IROOT.
`DOTRANS`	`false`	Transient spectra - starting from IROOT.
	`ALL`	Compute all possible transitions.

5.7 Excited States via ROCIS and ROCIS/DFT

The ORCA program package can perform configuration interaction with single excitations (CIS) calculations using a restricted open-shell Hartee-Fock (ROHF) reference function. It produces excitation energies, absorption energies and CD intensities. It was designed with the aim to reproduce and - even more importantly - reliably predict transition metal L-edges as observed in X-ray absorption spectroscopy (XAS).

The original ROCIS implementation [589], [590] is able to perform the CIS calculations only on top of a high-spin ROHF reference function. Starting in ORCA 6.0 the new General Spin ROCIS (GS-ROCIS) implementation [591] is also available upon request by the user. This new implementation can perform the CIS calculation on top of any spin coupling situation obtained with the CSF-ROHF method. This includes the high-spin situations covered previously by the original implementation.

5.7.1 General Description

General Spin ROCIS

The GS-ROCIS method is described as follows:

Starting from a reference CSF constructed in the genealogical coupling scheme (obtained by a `CSF-ROHF` calculation), four classes of excited CSFs are defined according to the occupation number of the orbitals involved in the excitation.

These excitation classes are:

- DOMO to SOMO excitations - $|\Phi_i^t\rangle$
- SOMO to VMO excitations - $|\Phi_t^a\rangle$
- DOMO to VMO excitations - $|\Phi_i^a\rangle$

- DOMO to VMO excitations coupled with SOMO to SOMO excitations - $|\Phi_{ui}^{at}\rangle$

This way, the GS-ROCIS wavefunction is defined as:

$$\Psi_{GS-ROCIS} = c_0 |\Phi_0\rangle + \sum_{i,t} c_i^t |\Phi_i^t\rangle + \sum_{t,a} c_t^a |\Phi_t^a\rangle + \sum_{i,a} c_i^a |\Phi_i^a\rangle + \sum_{i,t,a,u\neq t} c_{ui}^{at} |\Phi_{ui}^{at}\rangle$$

Where the expansion coefficients c_p^q are determined in the CI procedure.

`GS-ROCIS` can be requested by:

```
%rocis
  DoGenROCIS true  # Turns the general-spin ROCIS procedure on
  ReferenceMult 1  # The reference wavefunction multiplicity (it needs to agree␣
↪with the ROHF solution)
end
```

A number of basic variables in the `%rocis` block control the settings of the Davidson procedure that is used to solve the CI problem:

```
%rocis
  NRoots    6     # Number of desired roots.
  MaxDim    5     # Davidson expansion space = MaxDim * NRoots.
  ETol      1e-6  # Energy convergence tolerance.
  RTol      1e-6  # Residual vector convergence tolerance.
  MaxIter   35    # Maxmimum number of iterations.
  NGuessMat 512   # Dimension of the guess matrix: 512x512.
end
```

The amount of output produced during the calculation is controlled via the `PrintLevel` variable:

```
%rocis
  PrintLevel 3
end
```

Note, that this does not influence which spectra are calculated or printed.

The `orca_rocis` module provides two ways of choosing the orbital excitation space: by orbital energy or orbital number. In the former case an energy window has to be specified and the program will then take all orbitals, whose orbital energies lie within this window, into account. Note, that one actually has to define two orbital windows: One for the donor and the second for the acceptor orbital. The input of the windows is done as an array: The first two numbers define the donor space while the last two numbers define the acceptor space.

```
%rocis
  NRoots 3
  EWin = -5,5,-5,5
end
```

The default is to keep core orbitals and very high lying virtual orbitals out of their respective orbital excitation spaces. Since these orbitals span a space that is usually not reachable with regular UV/Vis spectroscopy, this is a reasonable approximation. One has to keep in mind that an orbital energy window makes only sense if the orbitals used in the calculation have a well-defined orbital energy. As a consequence one cannot use an orbital energy window for a calculation with localized orbitals. The second way to specify the excitation space is by orbital numbering.

```
%rocis
  NRoots 3
  OrbWin = 1,13,9,22
end
```

In restricted calculations only one set of spatial orbitals is created. Hence it is not necessary to provide orbital windows for α and β electrons separately. Of course, only doubly or singly occupied orbitals can act as donor orbitals and only singly and nonoccupied orbitals can act as acceptor orbitals. The program recognises nonoccupied orbitals in the donor space and doubly occupied orbitals in the acceptor space and removes both.

The absorption spectrum calculated on the basis of the pure dipole approximation for your calculation is always printed.

Furthermore like in TD-DFT (section *Excited States via RPA, CIS, TD-DFT and SF-TDA*) or CASSCF one may obtain intensities by evaluating the 2nd order oscillation strengths, or the full semi-classical oscillation strengths.

- The exact oscillation strengths behave like the multipole expansion in the velocity representation.
- They are by definition origin independent they do not suffer from artificial negative values like the multipole moments beyond 1st order.
- They are used with the multipole moments up to 2nd order to regenerate the electric dipole, electric quadrupole and magnetic dipole contributions in either length or the velocity representation.

```
%rocis
  DoDipoleLength        true
  DoDipoleVelocity      true
  DecomposeFosc         true
  DoFullSemiclassical   true
  DoCD                  true
end
```

Further details can be found in the section *One Photon Spectroscopy*.

The user can control which orbital excitations to include in the ROCIS problem:

```
%rocis
  Do_is true     # Include DOMO->SOMO excitations
  Do_sa true     # Include SOMO->Virtual excitations
  Do_ia true     # Include DOMO->Virtual excitations
  Do_ista true   # Include DOMO->SOMO coupled to
                 # SOMO->Virtual excitations with s not equal t

  # For GS-ROCIS, the following excitation is included in the DOMO->VMO class
  # Only on the original ROCIS implementation it is a separate class.
  Do_isa true    # Include DOMO->SOMO coupled to
                 # SOMO->Virtual excitations with s = t
end
```

By default, all excitations are included.

When the program finishes the ROCIS calculation it gives the excitation energy together with the composition for each root and the respective spin coupling situation. According to the number of labels of the respective functions $|\Phi\rangle$, contributions from excited configuration state functions belonging to the different excitation classes are given by two, three or four numbers.

```
STATE  16   Exc. Energy: 27294.664mEh  742.726eV     5990486.3cm**-1
       8->50                 :   0.0145   (-0.120499) : spin coupling: +1-1+1+1+1+1
       8->51                 :   0.0317   (0.178098)  : spin coupling: +1+1+1+1+1-1
       8->51                 :   0.2212   (-0.470333) : spin coupling: +1+1-1+1+1+1
       8->51                 :   0.4135   (-0.643076) : spin coupling: +1-1+1+1+1+1
       6->46   ;   48->51    :   0.0843   (-0.290338) : spin coupling: +1 2+1 0+1+1
       6->47   ;   49->51    :   0.1252   (-0.353805) : spin coupling: +1+1 2+1 0+1
       7->46   ;   49->51    :   0.0452   (-0.212665) : spin coupling: +1 2+1+1 0+1
       7->47   ;   48->51    :   0.0133   (0.115519)  : spin coupling: +1+1 2 0+1+1
       7->49   ;   46->51    :   0.0109   (0.104242)  : spin coupling: +1 0+1+1 2+1
```

Furthermore the ROCIS is able to calculate the effect of spin-orbit coupling (SOC) on the calculated ground and excited states. It introduces SOC in the framework of quasi-degenerate perturbation theory (QDPT). The SOC Hamiltonian is diagonalized in the basis of the calculated ROCIS states $\left|\Psi_I^{SM}\right\rangle$, where I is the root label and S and M are the spin and magnetic spin quantum numbers, respectively[656], [589].

```
%rocis
  rel
```

(continues on next page)

(continued from previous page)

```
    DoSOC true        # invokes the calculation of SOC effects
    TEMPERATURE 10    # temperature for SOC corrected spectra in Kelvin
  end
end
```

After the SOC calculation the program will produce additional spectra for the SOC corrected results. The spectra contain transitions from the $2S+1$ lowest lying states into all excited states, where S is the spin quantum number of the electronic ground state. These $2S+1$ lowest states may be split up in the order of 1-100 cm^{-1}. Due to the small magnitude of the splitting, all of the $2S+1$ states can be significantly populated even at low temperatures. Experimentally, the intensity of a given transition is dependent on the population of the corresponding initial state. With the `TEMPERATURE` keyword the population of the theoretically calculated states can be manipulated by the varying the fictive temperature of the system. It has to be mentioned that the electric quadrupole transitions between spin-orbit coupled states are not well defined and are likely to give unreasonable results. Hence it is recommended to use the `Decomposefosc` keyword only for calculations that do not include SOC.

```
------------------------------------------------------------------------------
↪---------------------
      SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
------------------------------------------------------------------------------
↪---------------------
      Transition          Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                           (eV)      (cm-1)    (nm)  (*population)  (au**2)    (au)␣
↪     (au)      (au)
------------------------------------------------------------------------------
↪---------------------
  0-5.0A  ->  1-5.0A    0.000000       0.0     0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-5.0A  ->  3-5.0A    0.000000       0.0  2927278012.6   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  3-5.0A  ->  4-5.0A    0.000001       0.0  2013441147.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  2-5.0A  ->  4-5.0A    0.000001       0.0  1192923906.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  2-5.0A    0.000002       0.0  559585326.1   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  2-5.0A    0.000002       0.0  554991701.4   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  3-5.0A    0.000003       0.0  469780906.9   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  3-5.0A    0.000003       0.0  466539107.5   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  4-5.0A    0.000003       0.0  380906816.8   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  4-5.0A    0.000003       0.0  378772788.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  4-5.0A  ->  5-5.0A   715.072294 5767447.6     1.7   0.000853852   0.00024   0.
↪00000   0.01494   0.00454
  3-5.0A  ->  5-5.0A   715.072294 5767447.6     1.7   0.000001143   0.00000   0.
↪00021   0.00018   0.00050
  2-5.0A  ->  5-5.0A   715.072295 5767447.6     1.7   0.000001245   0.00000   0.
↪00024   0.00053   0.00012
  1-5.0A  ->  5-5.0A   715.072297 5767447.6     1.7   0.000000330   0.00000   0.
↪00000   0.00029   0.00010
  0-5.0A  ->  5-5.0A   715.072297 5767447.6     1.7   0.000024144   0.00001   0.
↪00000   0.00251   0.00076
  4-5.0A  ->  6-5.0A   715.089740 5767588.3     1.7   0.000002661   0.00000   0.
↪00067   0.00036   0.00042
  3-5.0A  ->  6-5.0A   715.089741 5767588.3     1.7   0.000008083   0.00000   0.
↪00000   0.00145   0.00045
```

(continues on next page)

(continued from previous page)

```
  2-5.0A  ->  6-5.0A   715.089742 5767588.3     1.7   0.000775076   0.00022   0.
↪00001   0.01423   0.00432
  1-5.0A  ->  6-5.0A   715.089744 5767588.4     1.7   0.000000497   0.00000   0.
↪00002   0.00015   0.00035
  0-5.0A  ->  6-5.0A   715.089744 5767588.4     1.7   0.000001537   0.00000   0.
↪00056   0.00035   0.00007
  4-5.0A  ->  7-5.0A   715.115140 5767793.2     1.7   0.000001500   0.00000   0.
↪00027   0.00020   0.00056
  3-5.0A  ->  7-5.0A   715.115141 5767793.2     1.7   0.000834113   0.00024   0.
↪00001   0.01476   0.00449
  2-5.0A  ->  7-5.0A   715.115141 5767793.2     1.7   0.000008694   0.00000   0.
↪00002   0.00150   0.00050
  1-5.0A  ->  7-5.0A   715.115143 5767793.2     1.7   0.000000562   0.00000   0.
↪00026   0.00025   0.00018
  0-5.0A  ->  7-5.0A   715.115143 5767793.2     1.7   0.000003341   0.00000   0.
↪00036   0.00031   0.00085
  4-5.0A  ->  8-5.0A   715.184940 5768356.2     1.7   0.000000146   0.00000   0.
↪00004   0.00020   0.00001
  3-5.0A  ->  8-5.0A   715.184940 5768356.2     1.7   0.000019943   0.00001   0.
↪00206   0.00012   0.00120
...
```

If the `PrintLevel` value is set to 3 or higher, the program will print out the composition of the SOC corrected states in the basis of states $\left|\Psi_I^{SM}\right\rangle$.

```
Eigenvectors of SOC calculation:
the threshold for printing is: 0.010000
          weight          real           image     :   Root   Block   Spin    Ms
State 0:            0.00 cm**-1          0.00000 eV
        0.135363      -0.028407       0.366819   :      0      0      2      2
        0.181749       0.224125       0.362653   :      0      0      2      1
        0.367330      -0.135925      -0.590639   :      0      0      2      0
        0.180828       0.039792      -0.423372   :      0      0      2     -1
        0.134728       0.185875       0.316510   :      0      0      2     -2

State 1:            0.00 cm**-1          0.00000 eV
        0.078367      -0.271364       0.068762   :      0      0      2      2
        0.306783       0.460606       0.307612   :      0      0      2      1
        0.228162       0.440545       0.184613   :      0      0      2      0
        0.307656      -0.543907      -0.108722   :      0      0      2     -1
        0.079031      -0.142362      -0.242414   :      0      0      2     -2

State 2:            0.02 cm**-1          0.00000 eV
        0.416636       0.252581       0.594003   :      0      0      2      2
        0.072004      -0.082420       0.255363   :      0      0      2      1
        0.022669      -0.134504       0.067662   :      0      0      2      0
        0.071994       0.254211       0.085850   :      0      0      2     -1
        0.416695      -0.326890      -0.556631   :      0      0      2     -2

State 3:            0.02 cm**-1          0.00000 eV
        0.080023      -0.256741       0.118772   :      0      0      2      2
        0.319001       0.556244       0.097947   :      0      0      2      1
        0.201858      -0.134225       0.428768   :      0      0      2      0
        0.319074       0.512787       0.236903   :      0      0      2     -1
        0.080042       0.143270       0.243959   :      0      0      2     -2

State 4:            0.03 cm**-1          0.00000 eV
        0.289609      -0.315623      -0.435879   :      0      0      2      2
        0.120461       0.303600      -0.168190   :      0      0      2      1
        0.179979      -0.232115      -0.355108   :      0      0      2      0
        0.120448       0.275975      -0.210441   :      0      0      2     -1
```

(continues on next page)

(continued from previous page)

```
        0.289502      -0.272469      -0.463963   :      0     0     2     -2

State 5:     5767447.63 cm**-1       715.07230 eV
        0.191190      -0.086278       0.428656   :      1     0     2      2
        0.061035      -0.247045      -0.001820   :      1     0     2      1
        0.055460      -0.038920       0.232262   :      1     0     2      0
        0.061035      -0.234143      -0.078813   :      1     0     2     -1
        0.191190      -0.058171       0.433366   :      1     0     2     -2
        0.109587      -0.316206      -0.097982   :      2     0     2      2
        0.026192      -0.108605      -0.119987   :      2     0     2      1
        0.010968       0.017308      -0.103287   :      2     0     2      0
        0.026192      -0.141786       0.078029   :      2     0     2     -1
        0.109587       0.330874       0.010450   :      2     0     2     -2
        0.015452       0.114128      -0.049265   :      3     0     2      2
        0.041882      -0.204634      -0.002607   :      3     0     2      1
        0.027697       0.027504      -0.164135   :      3     0     2      0
        0.041882      -0.194305      -0.064244   :      3     0     2     -1
        0.015452      -0.091834      -0.083778   :      3     0     2     -2

State 6:     5767588.34 cm**-1       715.08974 eV
        0.198585      -0.131642       0.425741   :      1     0     2      2
        0.047453      -0.161434       0.146261   :      1     0     2      1
        0.019867      -0.139025      -0.023213   :      1     0     2      0
        0.047453       0.105160       0.190773   :      1     0     2     -1
        0.198585       0.013812      -0.445415   :      1     0     2     -2
        0.160559      -0.388214      -0.099243   :      2     0     2      2
        0.044707      -0.072788      -0.198517   :      2     0     2      1
        0.014352      -0.118163      -0.019729   :      2     0     2      0
        0.044707       0.133333      -0.164102   :      2     0     2     -1
        0.160559      -0.399398      -0.032262   :      2     0     2     -2
        0.012669      -0.111020      -0.018537   :      3     0     2      0

...
```

Further details of the SOC calculation such as the procedure of SOC integral calculation can be controlled via the `%rel` block (section *Relativistic Calculations*.

Original ROCIS Implementation

It is still possible to use the original implementation of ROCIS[589], [590] which is able to perform CIS calculations only on top of a high-spin ROHF reference function. All spins of the unpaired electrons have to be coupled ferromagnetically to give a total spin of $S = \frac{1}{2}N$, where N is the number of unpaired electrons.

The usage of the original ROCIS implementation can be requested by:

```
%rocis
  DoGenROCIS false
end
```

In our experience ROHF calculations suffer a lot from convergence problems. UHF calculations generally exhibit better convergence properties. In most cases the quasi-restricted orbitals (qro's) of a UHF calculation resemble the ROHF orbitals. Thus the original ROCIS features the ability to start a calculation on top of a UHF calculation. It will automatically create the qro's and build the reference determinant with them. If one wants to avoid the (small) errors that are introduced by this procedure, one may take the qro's of a UHF calculation as starting orbitals for a subsequent ROHF calculation.

ROCIS/DFT

Important

- Currently, `ROCIS/DFT` is not implemented for the general-spin (`GS-ROCIS`) procedure.

For many transition metal compounds the description of the electronic ground and excited states by Hartree-Fock theory and CIS is of rather poor quality. Especially covalency and relative spin state energetics are not reproduced correctly. This in turn might lead to wrong intensity distributions in the calculated L-edge spectra. In the majority of these cases the quality of the description and hence the predicted L-edge spectra can be significantly improved with the ROCIS/DFT method[589]. It features the usage of a restricted open-shell Kohn-Sham matrix as reference and also uses the DFT orbitals for setting up the excited configuration state functions in the CI expansion. The two electron integrals that include the DFT orbitals are scaled according to their nature and their position in the CI matrix by the parameters c_1, c_2 and c_3. They all lie in the interval [0;1]. Parameters c_1 and c_2 scale coulomb- and exchange-like terms in the diagonal part of the CI matrix, whereas c_3 reduces the size of all off-diagonal elements of the CI matrix. For example:

$$\begin{aligned} H_{ia,ia}^{\text{DFT/ROCIS}} &= F_{aa}^{C(\text{KS})} - F_{ii}^{C(\text{KS})} - c_1\,(ii|aa) + 2c_2\,(ia|ia) \\ H_{ia,jb}^{\text{DFT/ROCIS}} &= c_3\left\{\delta_{ij}F_{ab}^{C(\text{KS})} - \delta_{ab}F_{ji}^{C(\text{KS})} - (ij|ab) + 2\,(ia|jb)\right\} \end{aligned} \tag{5.83}$$

The three default parameters $c_1 = 0.18$, $c_2 = 0.20$ and $c_3 = 0.40$ have been optimized for a test set of molecules and their excited states on a B3LYP/def2-TZVP(-f) level of theory but can be freely chosen[589]. It is most likely that for a different combination of test molecules, functional and basis set, a different set of parameters gives better results. Since the parameters are chosen with regard of a good "balance" between orbital energies, Coulomb and exchange integrals, a new set of parameters should at least crudely resemble their relative proportions.

The ROCIS/DFT method can be requested by the following keywords:

```
%rocis
  DoGenROCIS            false   # ROCIS/DFT is available only for the high-spin␣
↪implementation of ROCIS.
  DoDFTCIS               true   # Switches on the ROCIS/DFT method.
  DFTCIS_c = 0.18, 0.20, 0.40   # Array input of the three parameters.
end
```

Resolution of the Identity

If calculations on large molecules are conducted, the integral transformation will be the most time-consuming part. Therefore it is strongly recommended to use the resolution of the identity (RI) approximation in those cases. It effectively reduces the computational costs of the transformation step by only introducing minor errors to the calculation. It has to be kept in mind that in order to keep the introduced errors small, one has to provide a reasonable auxiliary basis sets along with your normal basis set input.

```
! def2-SVP def2-SVP/C TightSCF

%scf
  HFTyp ROHF
  ROHF_Case HighSpin
  ROHF_Nel[1] = 1
end

%rocis
  NRoots 5
  DoRI true   # invokes the RI approximation
end

* xyz 0 2
```

(continues on next page)

(continued from previous page)

```
 N 0 0 0
 O 0 0 1.15
*
```

5.7.2 Core PNO-ROCIS, PNO-ROCIS/DFT

It has been shown [657] that it is possible to combine the powerful machinery of Pair Natural Orbitals (PNOs) with the ROCIS and ROCIS/DFT methods to formulate the core PNO-ROCIS and PNO-ROCIS/DFT methods. The usage of PNOs here is somewhat unconventional since they are not used to treat electron correlation effects in a state specific manner. Rather, the PNOs are used to identify the relevant part of the virtual space that can be reached by excitation out of local core orbitals. This subspace of the virtual space is local, thus leading to a linear scaling, state universal method.

The PNO-ROCIS calculations can be requested with the following keywords:

```
%rocis
  DoPNO     true       # Flag to call the PNO truncation
  TCutPNO   1e-11      # Threshold to cutout the PNO populations
  XASElems  0          # Number of the involved element to the calculated core XAS␣
↪calculation
end
```

As shown in reference[657], a universal TCutPNO 1e-11 threshold can be defined for all edges provided that the PNOs are constructed by taking into account all the available core orbitals in the systems. For example in the case of a 1st row transition metal this will be the 9 1s, 2s, 2p, 3s and 3p MOs. These orbitals will be identified automatically by the program provided that the elements for which the XAS calculation will be performed are specified within the XASElems keyword.

```
=================================================================
Core PNO/ROCIS truncation
=================================================================

...

=================================================================
TCutPNO:                              1.000e-13
Virtual orbitals before selection:   303 ... 2772 (2470 MO's)
Virtual orbitals after selection:    303 ...  558 (256 MO's)
PNO transformation completed in:    3253.03 sec
=================================================================
```

From this point and on the programm will proceed the usual way. This will result in extraordinary computation speeding ups without loss in accuracy.

5.7.3 Transition Metal L-Edges with ROCIS or DFT/ROCIS

The ROCIS method was designed to calculate transition metal L-edge spectra of large molecules as they are observed in X-ray absorption spectroscopy (XAS). An L-edge results when an electron is promoted from the 2p shell of a transition metal ion into the valence d shell by an X-ray photon. Strong spin-orbit coupling in the 2p shell and p-d coupling phenomena complicate the interpretation and even more so the prediction of these spectra. It has to be kept in mind that the present program applies a variety of approximations which might lead to observable deviations from experimentally determined spectra. However, we believe that the results obtained from the program are in general qualitatively correct and in most cases accurate close to the experimental uncertainty. In cases where quantitative accuracy is not met, the provided results might still give some insight into the mechanisms of intensity distribution in the spectra.

The special input structure for orbital windows described in *Excited States with Restricted Open-shell CIS - ROCIS* allows the user to restrict the donor orbital space to the transition metal 2p shell. The acceptor orbital space is the

same as in regular UV/Vis spectroscopy. It should include all singly occupied molecular orbitals and as many virtual orbitals as one can afford in the calculation (preferably all). The number of roots should be chosen large enough so that at least all 2p-3d single excitations are calculated. In many cases even more roots are required since doubly excited or charge transfer states may become important. Moreover the strong SOC apparent in the 2p shell of transition metal ions necessitates the additional calculation of excited states with a total spin of $S' = S+1$ and $S' = S-1$ where S is the total spin of the electronic ground state. Accordingly, additional excitation classes introduce excited configuration state functions with a lower and higher spin multiplicity.

for $S' = S + 1$:

- DOMO to SOMO excitations - $\left|\Phi_i^{t+}\right\rangle$
- SOMO to VMO excitations - $\left|\Phi_t^{a+}\right\rangle$
- DOMO to VMO excitations - $\left|\Phi_i^{a+}\right\rangle$

for $S' = S - 1$:

- DOMO to SOMO excitations - $\left|\Phi_i^{t-}\right\rangle$
- SOMO to VMO excitations - $\left|\Phi_t^{a-}\right\rangle$
- DOMO to VMO excitations - $\left|\Phi_i^{a-}\right\rangle$
- SOMO to SOMO excitations - $\left|\Phi_t^{u-}\right\rangle$

Note

- Depending on the ground state multiplicity and/or occupation of the involved orbitals, some excitations are not possible. The program automatically includes only the allowed excitations into the respective multiplicities.

Inclusion of configuration state functions with higher or lower multiplicity is invoked with the keywords `DoLowerMult` and `DoHigherMult`, respectively.

```
%rocis
  DoLowerMult true     #Invokes a CI calculation  #with S'=S-1
  DoHigherMult true    #Invokes a CI calculation  #with S'=S+1
end
```

The program will conduct a separate Davidson procedure for each multiplicity. Subsequently it gives the excitation energies and compositions of the calculated excited states for all included multiplicities.

Here is an example calculation of L-edge XAS on $[FeCl_4]^{2-}$:

```
# [Fe(II)Cl4]2- - ROCIS - L-edge

!x2c x2c-SVPall AutoAux TightSCF

%pal
  nproc 12
end

%scf
  HFTyp ROHF
  ROHF_CASE HIGHSPIN
  ROHF_NEL[1] 4
end

%rocis
  DoGenROCIS true
  ReferenceMult 5
  NRoots 50
```

(continues on next page)

(continued from previous page)

```
  OrbWin 6,8,0,2000
  DoPNO true
  TCutPNO 1e-11
  XASelems 0
  rel
    DoSOC true
  end
  DoHigherMult true
  DoLowerMult true
  DoRI true
  Decomposefosc true
end

*xyz -2 5
  Fe    -0.009575    0.000087    0.011550
  Cl    -1.774578   -1.558533   -0.219791
  Cl    -0.681385    1.874633    1.289404
  Cl     0.666770    0.847828   -2.091305
  Cl     1.756927   -1.164015    1.071196
end
```

After all CI calculations are finished, the program gives a list of all calculated roots with their excitation energies and their multiplicities. It is this number that will be referred to as label I in the decomposition of spin-orbit coupled states in the basis $\left|\Psi_I^{SM}\right\rangle$. It is very important to note, that when states with different multiplicities are calculated this number might deviate from the number that appears in the respective CI part of the output. If one gets confused about the numbering of the states, the state energies might act as a guideline through the output of the program.

```
-------------------
Excitation energies
-------------------


--------------------------------------------------------------------------------
    ROOT    Mult      Excitation energy[Eh]           [cm-1]           [eV]
--------------------------------------------------------------------------------
     0       5            0.00000000                     0.00           0.000
     1       5            26.41778649                5798033.94         718.865
     2       5            26.42452401                5799512.66         719.048
     3       5            26.42621555                5799883.91         719.094
     4       5            26.44615744                5804260.65         719.637
     5       5            26.45323131                5805813.18         719.829
     6       5            26.47192609                5809916.21         720.338
     7       5            26.47357807                5810278.78         720.383
     8       5            26.48177016                5812076.74         720.606
     9       5            26.49063895                5814023.21         720.847
    10       5            26.50722706                5817663.88         721.298
    11       5            26.50759779                5817745.25         721.308
    12       5            26.50982238                5818233.49         721.369
    13       3            26.55037043                5827132.76         722.472
    14       3            26.58407940                5834531.02         723.390
    15       3            26.58587963                5834926.12         723.439
    16       3            26.58857702                5835518.13         723.512
    17       3            26.59674863                5837311.59         723.734
    18       3            26.59825865                5837643.00         723.775
    19       3            26.61152227                5840554.03         724.136
    20       3            26.61806435                5841989.85         724.314
    21       3            26.62812795                5844198.56         724.588
    22       3            26.63079807                5844784.58         724.661
    23       3            26.63345218                5845367.09         724.733
    24       3            26.64337021                5847543.85         725.003
    25       3            26.65680719                5850492.92         725.369
```

(continues on next page)

(continued from previous page)

```
    26       3            26.66486245                5852260.85        725.588
    27       3            26.66512507                5852318.49        725.595
    28       3            26.66741755                5852821.63        725.657
    29       3            26.68096811                5855795.63        726.026
    30       3            26.68291870                5856223.74        726.079
    31       3            26.68410859                5856484.89        726.112
    32       3            26.68502036                5856685.00        726.136
    33       3            26.68625712                5856956.44        726.170
    34       3            26.68835097                5857415.98        726.227
    35       3            26.69951161                5859865.46        726.531
    36       3            26.70277202                5860581.04        726.619
    37       3            26.71055426                5862289.04        726.831
    38       3            26.71604333                5863493.75        726.980
    39       3            26.71631714                5863553.85        726.988
    40       3            26.71801283                5863926.01        727.034
    41       3            26.71869535                5864075.80        727.053
    42       3            26.71953590                5864260.28        727.076
    43       3            26.72200411                5864801.99        727.143
    44       3            26.72811146                5866142.40        727.309
    45       3            26.73591005                5867853.99        727.521
    46       3            26.73675062                5868038.48        727.544
    47       3            26.75216877                5871422.37        727.964
    48       3            26.78204828                5877980.17        728.777
    49       7            27.21891306                5973860.90        740.664
    50       7            27.21926225                5973937.54        740.674
    51       7            27.21931555                5973949.24        740.675
    52       5            27.26974929                5985018.16        742.048
    53       5            27.26978849                5985026.77        742.049
    54       5            27.27023065                5985123.81        742.061
    55       5            27.29466376                5990486.26        742.726
    56       5            27.29677691                5990950.04        742.783
    57       5            27.31333680                5994584.52        743.234
    58       7            27.31641173                5995259.39        743.317
    59       5            27.31967798                5995976.24        743.406
    60       7            27.32002169                5996051.68        743.416
...
```

Without SOC the spin exclusion rule applies which means that only excited states with a total spin equal to the ground state spin ($S' = S$) give rise to non-vanishing intensities. Hence, only these transitions are listed in the spectra before SOC.

```
------------------------
ROCIS-EXCITATION SPECTRA
------------------------

Center of mass = ( -0.0160,  0.0000,  0.0229)
Reading integrals                                                ...
Reading the Dipole integrals                                     ... done
Reading the Linear Momentum integrals                            ... done
Reading the Angular momentum integrals                           ... done
Reading the Quadrupole integrals                                 ... done
Reading the Higher Moment integrals                              ... done
-------------------------------------------------------------------
Using One-Photon Spectroscopy Tool
-------------------------------------------------------------------


-------------------------------------------------------------------------------
 ↪-----------------
                     ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
 ↪-----------------
```

(continues on next page)

(continued from previous page)

```
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX      ␣
↪  DY        DZ
                      (eV)       (cm-1)    (nm)                 (au**2)    (au)     ␣
↪ (au)       (au)
-------------------------------------------------------------------------------
↪-----------------
  0-5A  ->  1-5A   718.864517 5798034.0     1.7   0.007967331   0.00045   0.00001 ␣
↪-0.02035   0.00619
  0-5A  ->  2-5A   719.047854 5799512.7     1.7   0.000032204   0.00000   0.00123 ␣
↪-0.00019  -0.00052
  0-5A  ->  3-5A   719.093883 5799883.9     1.7   0.000038607   0.00000  -0.00060 ␣
↪-0.00039  -0.00130
  0-5A  ->  4-5A   719.636530 5804260.7     1.7   0.000000074   0.00000  -0.00004 ␣
↪-0.00001  -0.00005
  0-5A  ->  5-5A   719.829019 5805813.2     1.7   0.000257559   0.00001  -0.00007 ␣
↪-0.00364   0.00116
  0-5A  ->  6-5A   720.337730 5809916.2     1.7   0.045609938   0.00258  -0.04638 ␣
↪ 0.00619   0.01987
  0-5A  ->  7-5A   720.382683 5810278.8     1.7   0.043374431   0.00246   0.02031 ␣
↪ 0.01315   0.04327
  0-5A  ->  8-5A   720.605601 5812076.7     1.7   0.000000543   0.00000   0.00005 ␣
↪ 0.00005   0.00016
  0-5A  ->  9-5A   720.846933 5814023.2     1.7   0.180244898   0.01021   0.00031 ␣
↪ 0.09663  -0.02948
  0-5A  -> 10-5A   721.298318 5817663.9     1.7   0.084488140   0.00478  -0.02823 ␣
↪-0.01833  -0.06040
  0-5A  -> 11-5A   721.308407 5817745.3     1.7   0.085964316   0.00486   0.06366 ␣
↪-0.00850  -0.02719
  0-5A  -> 12-5A   721.368941 5818233.5     1.7   0.000000007   0.00000  -0.00000 ␣
↪ 0.00002   0.00001
  0-5A  -> 13-5A   742.047603 5985018.2     1.7   0.002287503   0.00013   0.00507 ␣
↪ 0.00276   0.00962
  0-5A  -> 14-5A   742.048670 5985026.8     1.7   0.002278095   0.00013   0.00999 ␣
↪-0.00164  -0.00477
  0-5A  -> 15-5A   742.060702 5985123.8     1.7   0.001342355   0.00007   0.00019 ␣
↪ 0.00823  -0.00246
  0-5A  -> 16-5A   742.725560 5990486.3     1.7   0.000418059   0.00002   0.00197 ␣
↪ 0.00126   0.00419
  0-5A  -> 17-5A   742.783062 5990950.1     1.7   0.000464555   0.00003   0.00461 ␣
↪-0.00063  -0.00197
  0-5A  -> 18-5A   743.233680 5994584.5     1.7   0.000000003   0.00000  -0.00001 ␣
↪-0.00000  -0.00001
  0-5A  -> 19-5A   743.406232 5995976.3     1.7   0.000000000   0.00000  -0.00000 ␣
↪-0.00000  -0.00000
  0-5A  -> 20-5A   743.562910 5997239.9     1.7   0.000577479   0.00003   0.00001 ␣
↪ 0.00538  -0.00165
  0-5A  -> 21-5A   743.597306 5997517.4     1.7   0.000009265   0.00000  -0.00029 ␣
↪-0.00017  -0.00063
  0-5A  -> 22-5A   743.658216 5998008.6     1.7   0.000010720   0.00000  -0.00070 ␣
↪ 0.00011   0.00029
  0-5A  -> 23-5A   744.043391 6001115.3     1.7   0.000081920   0.00000  -0.00088 ␣
↪-0.00059  -0.00184
  0-5A  -> 24-5A   744.092953 6001515.0     1.7   0.000301862   0.00002  -0.00017 ␣
↪-0.00389   0.00119
...
```

After calculation of SOC in the basis of all calculated ROCIS roots, the program prints out the composition of the spin-orbit coupled states (if `PrintLevel` >2) and the corresponding absorption spectrum.

```
Eigenvectors of SOC calculation:
the threshold for printing is: 0.010000
```

(continues on next page)

(continued from previous page)

```
          weight         real           image    :   Root   Block   Spin    Ms
State 0:           0.00 cm**-1         0.00000 eV
        0.135363      -0.028407       0.366819   :      0     0     2      2
        0.181749       0.224125       0.362653   :      0     0     2      1
        0.367330      -0.135925      -0.590639   :      0     0     2      0
        0.180828       0.039792      -0.423372   :      0     0     2     -1
        0.134728       0.185875       0.316510   :      0     0     2     -2

State 1:           0.00 cm**-1         0.00000 eV
        0.078367      -0.271364       0.068762   :      0     0     2      2
        0.306783       0.460606       0.307612   :      0     0     2      1
        0.228162       0.440545       0.184613   :      0     0     2      0
        0.307656      -0.543907      -0.108722   :      0     0     2     -1
        0.079031      -0.142362      -0.242414   :      0     0     2     -2

State 2:           0.02 cm**-1         0.00000 eV
        0.416636       0.252581       0.594003   :      0     0     2      2
        0.072004      -0.082420       0.255363   :      0     0     2      1
        0.022669      -0.134504       0.067662   :      0     0     2      0
        0.071994       0.254211       0.085850   :      0     0     2     -1
        0.416695      -0.326890      -0.556631   :      0     0     2     -2

State 3:           0.02 cm**-1         0.00000 eV
        0.080023      -0.256741       0.118772   :      0     0     2      2
        0.319001       0.556244       0.097947   :      0     0     2      1
        0.201858      -0.134225       0.428768   :      0     0     2      0
        0.319074       0.512787       0.236903   :      0     0     2     -1
        0.080042       0.143270       0.243959   :      0     0     2     -2

State 4:           0.03 cm**-1         0.00000 eV
        0.289609      -0.315623      -0.435879   :      0     0     2      2
        0.120461       0.303600      -0.168190   :      0     0     2      1
        0.179979      -0.232115      -0.355108   :      0     0     2      0
        0.120448       0.275975      -0.210441   :      0     0     2     -1
        0.289502      -0.272469      -0.463963   :      0     0     2     -2

State 5:     5767447.63 cm**-1       715.07230 eV
        0.191190      -0.086278       0.428656   :      1     0     2      2
        0.061035      -0.247045      -0.001820   :      1     0     2      1
        0.055460      -0.038920       0.232262   :      1     0     2      0
        0.061035      -0.234143      -0.078813   :      1     0     2     -1
        0.191190      -0.058171       0.433366   :      1     0     2     -2
        0.109587      -0.316206      -0.097982   :      2     0     2      2
        0.026192      -0.108605      -0.119987   :      2     0     2      1
        0.010968       0.017308      -0.103287   :      2     0     2      0
        0.026192      -0.141786       0.078029   :      2     0     2     -1
        0.109587       0.330874       0.010450   :      2     0     2     -2
        0.015452       0.114128      -0.049265   :      3     0     2      2
        0.041882      -0.204634      -0.002607   :      3     0     2      1
        0.027697       0.027504      -0.164135   :      3     0     2      0
        0.041882      -0.194305      -0.064244   :      3     0     2     -1
        0.015452      -0.091834      -0.083778   :      3     0     2     -2

State 6:     5767588.34 cm**-1       715.08974 eV
        0.198585      -0.131642       0.425741   :      1     0     2      2
        0.047453      -0.161434       0.146261   :      1     0     2      1
        0.019867      -0.139025      -0.023213   :      1     0     2      0
        0.047453       0.105160       0.190773   :      1     0     2     -1
        0.198585       0.013812      -0.445415   :      1     0     2     -2
        0.160559      -0.388214      -0.099243   :      2     0     2      2
        0.044707      -0.072788      -0.198517   :      2     0     2      1
```

(continues on next page)

(continued from previous page)

```
        0.014352      -0.118163      -0.019729   :      2     0     2      0
        0.044707       0.133333      -0.164102   :      2     0     2     -1
        0.160559      -0.399398      -0.032262   :      2     0     2     -2
        0.012669      -0.111020      -0.018537   :      3     0     2      0
...
```

```
Energy levels:  cm-1      eV      Boltzmann populations for T = 300.00  K

   0 :         0.000     0.0000         2.00e-01
   1 :         0.000     0.0000         2.00e-01
   2 :         0.018     0.0000         2.00e-01
   3 :         0.021     0.0000         2.00e-01
   4 :         0.026     0.0000         2.00e-01

-------------------------------------------------------------------------------
↪---------------------
     SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪---------------------
     Transition         Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                         (eV)      (cm-1)    (nm)  (*population)  (au**2)    (au)␣
↪     (au)      (au)
-------------------------------------------------------------------------------
↪---------------------
  0-5.0A  ->  1-5.0A    0.000000       0.0     0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-5.0A  ->  3-5.0A    0.000000       0.0  2927278012.6   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  3-5.0A  ->  4-5.0A    0.000001       0.0  2013441147.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  2-5.0A  ->  4-5.0A    0.000001       0.0  1192923906.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  2-5.0A    0.000002       0.0  559585326.1   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  2-5.0A    0.000002       0.0  554991701.4   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  3-5.0A    0.000003       0.0  469780906.9   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  3-5.0A    0.000003       0.0  466539107.5   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  1-5.0A  ->  4-5.0A    0.000003       0.0  380906816.8   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  0-5.0A  ->  4-5.0A    0.000003       0.0  378772788.3   0.000000000   0.00000  ␣
↪0.00000   0.00000   0.00000
  4-5.0A  ->  5-5.0A   715.072294 5767447.6     1.7   0.000853852   0.00024   0.
↪00000   0.01494   0.00454
  3-5.0A  ->  5-5.0A   715.072294 5767447.6     1.7   0.000001143   0.00000   0.
↪00021   0.00018   0.00050
  2-5.0A  ->  5-5.0A   715.072295 5767447.6     1.7   0.000001245   0.00000   0.
↪00024   0.00053   0.00012
  1-5.0A  ->  5-5.0A   715.072297 5767447.6     1.7   0.000000330   0.00000   0.
↪00000   0.00029   0.00010
  0-5.0A  ->  5-5.0A   715.072297 5767447.6     1.7   0.000024144   0.00001   0.
↪00000   0.00251   0.00076
  4-5.0A  ->  6-5.0A   715.089740 5767588.3     1.7   0.000002661   0.00000   0.
↪00067   0.00036   0.00042
  3-5.0A  ->  6-5.0A   715.089741 5767588.3     1.7   0.000008083   0.00000   0.
↪00000   0.00145   0.00045
  2-5.0A  ->  6-5.0A   715.089742 5767588.3     1.7   0.000775076   0.00022   0.
↪00001   0.01423   0.00432
```

(continues on next page)

(continued from previous page)

```
  1-5.0A  ->  6-5.0A   715.089744 5767588.4     1.7   0.000000497   0.00000   0.
↪00002   0.00015   0.00035
  0-5.0A  ->  6-5.0A   715.089744 5767588.4     1.7   0.000001537   0.00000   0.
↪00056   0.00035   0.00007
  4-5.0A  ->  7-5.0A   715.115140 5767793.2     1.7   0.000001500   0.00000   0.
↪00027   0.00020   0.00056
  3-5.0A  ->  7-5.0A   715.115141 5767793.2     1.7   0.000834113   0.00024   0.
↪00001   0.01476   0.00449
  2-5.0A  ->  7-5.0A   715.115141 5767793.2     1.7   0.000008694   0.00000   0.
↪00002   0.00150   0.00050
  1-5.0A  ->  7-5.0A   715.115143 5767793.2     1.7   0.000000562   0.00000   0.
↪00026   0.00025   0.00018
  0-5.0A  ->  7-5.0A   715.115143 5767793.2     1.7   0.000003341   0.00000   0.
↪00036   0.00031   0.00085
  4-5.0A  ->  8-5.0A   715.184940 5768356.2     1.7   0.000000146   0.00000   0.
↪00004   0.00020   0.00001
  3-5.0A  ->  8-5.0A   715.184940 5768356.2     1.7   0.000019943   0.00001   0.
↪00206   0.00012   0.00120
  2-5.0A  ->  8-5.0A   715.184941 5768356.2     1.7   0.000019954   0.00001   0.
↪00119   0.00064   0.00197
  1-5.0A  ->  8-5.0A   715.184943 5768356.2     1.7   0.000507851   0.00014   0.
↪00004   0.01152   0.00348
  0-5.0A  ->  8-5.0A   715.184943 5768356.2     1.7   0.000022305   0.00001   0.
↪00003   0.00240   0.00077
  4-5.0A  ->  9-5.0A   715.206052 5768526.4     1.7   0.000049980   0.00001   0.
↪00006   0.00360   0.00113
  3-5.0A  ->  9-5.0A   715.206053 5768526.4     1.7   0.000007590   0.00000   0.
↪00077   0.00040   0.00119
...
```

The obtained spectra can be prepared for ploting with the use of `orca_mapspc`. Check section *orca_mapspc* for details.

5.7.4 Resonant Inelastic Scattering Spectroscopy

General

Starting from ORCA version 4.0 ROCIS module can be used to calculate RIXS spectra

The present implementation is directly based on the Kramers Heisenerg Dirac (KDH) expression formula for near resonant and resonant conditions

$$|\alpha_{\rho\lambda}(E_{ex},E_{sc})|^2_{Total} = \sum_F \left| \sum_V \frac{\langle F|\, m_\rho \,|V\rangle \langle V|\, m_\lambda \,|I\rangle}{E_{VI} - E_{ex} - i\frac{1}{2}\Gamma_V} \right|^2 \left\{ \frac{\Gamma_F}{(E_{FV} - E_{ex} + E_{sc})^2 + \frac{1}{4}{\Gamma_F}^2} \right\}$$

$$|\alpha_{\rho\lambda}(E_{ex},E_{sc},V)|^2_{resonant} = \sum_F |\langle F|\, m_\rho \,|V\rangle|^2 |\langle V|\, m_\lambda \,|I\rangle|^2 f(E_{VI},E_{FV},E_{ex},E_{sc},\Gamma_V,\Gamma_F)$$

$$|\alpha_{\rho\lambda}(E_{VI},E_{sc})|^2_{Direct} = \sum_V |\alpha_{\rho\lambda}(E_{VI},E_{sc},V)|^2_{resonant}$$

The resonance scattering cross section for total and direct cases, averaged over all orientations of the molecule and integrated over all directions and polarizations of scattered radiation is given in equations:

$$\sigma^{Total}_{RXES}(E_{ex},E_{sc}) = \frac{8\pi E_{sc}^3 E_{ex}}{9c^4} \sum_{\rho,\lambda=x,y,z} |\alpha_{\rho\lambda}(E_{ex},E_{sc})|^2_{Total}$$

$$\sigma^{Direct}_{RXES}(E_{ex},E_{sc}) = \frac{8\pi E_{sc}^3 E_{ex}}{9c^4} \sum_{\rho,\lambda=x,y,z} |\alpha_{\rho\lambda}(E_{ex},E_{sc})|^2_{Direct}$$

Interference effects can be then derived in a straightforward way from equation:

$$\sigma_{RXES}^{interference}(E_{ex}, E_{sc}) = \sigma_{RXES}^{Total}(E_{ex}, E_{sc}) - \sigma_{RXES}^{Direct}(E_{ex}, E_{sc})$$

In order to access RIXS spectroscopy in the ROCIS module one needs in addition to specify a 2nd donor space. This is specified by defining an OrbWin array with 6 elements: The first four elements define the ranges of the two donor spaces while the last two elements the respective acceptor space range.

```
OrbWin = 0,0,2,4,45,60
```

An important difference with respect to the conventional ROCIS or DFT/ROCIS calculations is the fact that two donor spaces of very different energy ranges are involved (e.g. K-edge, L-edge) which requires to restrict somewhat the acceptor space and saturate it with as many states as possible.

The main calling commands in order to perform a RIXS calculation within both ROCIS and CASSCF blocks are the following:

- RIXS true. Similar to absorption spectroscopy, this requests the RIXS calculation to be performed based on the calculated non-relativistic ground state multiplicity States
- RIXSSOC true. By turning-on this flag the RIXS is calculated by taking in account the relativistically corrected Ms States.
- Elastic true. This flag indicates whether the resonant condition in which the initial and Final states coincide should be taken into account. Note that the intensity of this spectral feature might be overestimated as presently the non resonant terms are not treated

The respective ROCIS input reads then as follows:

```
!B3LYP def2-TZVP  SlowConv

%Basis
  AuxC "def2/J"
end

%ROCIS
  NRoots 200
  PrintLevel 3
  MaxCore 4000
  DoRI true
  DoHigherMult true
  SOC true
  RIXS true    # Request RIXS calculation (NoSOC)
  RIXSSOC true # Request RIXS calculation (with SOC)
  Elastic true # Request RIXS calculation (Elastic)
  DoDFTCIS true
  DFTCIS_c =0.18,0.20,0.40
  OrbWin = 2,4,25,33,0,100
end
* xyzfile 2 2 test.xyz
```

When running the calculation one can monitor if the requested NRoots were sufficient enough to select the states dominated by both the donor orbital spaces

```
-----------------------------------------------------------------------------
ROOT    Mult      Excitation energy[Eh]           [cm-1]          [eV]
-----------------------------------------------------------------------------
0       2            0.00000000                      0.00          0.000
1       2            0.06611737                  14511.08          1.799
2       2            0.07728471                  16962.03          2.103
3       2            0.07732428                  16970.72          2.104
...
84       2            33.75471831                7408304.35        918.513
```

(continues on next page)

(continued from previous page)

```
85       2            33.77073325                7411819.22        918.948
86       2            33.77076955                7411827.19        918.949
87       4            34.06882971                7477243.83        927.060
88       2            34.07021441                7477547.74        927.098
...
```

If that is not the case the respective RIXS calculations will not be performed and a Warning Message will be generated:

```
Making the RIXS files ...
WARNING!: Flag for RIXS property calculation was identified but
there is zero number of Intermediate  and/or Final states:
No Cross-Section properties will be evaluated ...Skipping this part
TIP: Increase the number of NRoots and/or decrease or increase
the acceptor orbital space
...Done
```

A successful run on the other hand will generate the following messages for RIXS and RIXSSOC calculations.

```
-----------------------------------------------------------------------------------
ROCIS RIXS SPECTRUM
-----------------------------------------------------------------------------------

Making the RIXS data files for Inelastic and Elastic Scattering
Ground State:                  1
Intermediate States:           21
Final States:                  59
The RIXS cross section will be generated from:
60 Ground-Final State Pairs and   21 Intermediate States/Pair
Calculating Intensities...
10% done
20% done
30% done
40% done
50% done
60% done
70% done
80% done
90% done
100% done
Storing the files...All Done
-----------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------------
ROCIS RIXSSOC SPECTRUM
-----------------------------------------------------------------------------------

Making the RIXS-SOC data files for Inelastic and Elastic Scattering
Ms States:                  2
Intermediate States:        78
Final States:               214
The RIXS cross section will be generated from:
432 Ground-Final State Pairs and   78 Intermediate States/Pair

Calculating Intensities...
10% done
20% done
30% done
40% done
50% done
60% done
70% done
```

(continues on next page)

(continued from previous page)

```
80% done
90% done
100% done

Storing the files...All Done
-------------------------------------------------------------------------------
```

In both cases the number of involved Initial, Final and Intermediate states is specified explicitly.

For example in the case of RIXSSOC 2 Ms Ground states, 78 Intermediate states and 214 Final states are involved. Then the RIXS cross section for elastic and inelastic scattering will be generated by 432 (2*(2+214)) Ground-Final State-Pairs and 78 Intermediate States per Ground-Final state pair.

Processing the spectra with `orca_mapspc`

By calling `orca_mapspc` with the following keywords:

```
orca_mapspc test.el_inel.rocis.rixssoc RIXS -x0871 -x1876 -x2-1 -x34 -w0.4 -g0.4
        -l -n125 -m125 -dx20 -eaxis1
```

The program will process the `test.el_inel.rocis.rixssoc` file with the following parameters:

Energy range along x : 871-876 eV

Energy range along y: -1-34 eV

-l indicates Lorentzian broadening

Width along x (gamma): 0.4 eV

Width along y (gamma): 0.4 eV

Points along x: 125

Points along y:125

Shift to be applied along Incident energy/Emission axis: 20 eV

The y axis will be Energy Transfer axis. If -eaxis2 is the y axis will be then Emission Energy axis

All this information is printed during the data processing:

```
Mode is RIXS
Using Lorentzian shape
Cannot read the paras.inp file ...
taking the line width parameter from the command line
Cannot read the udex.inp file ...
taking the excitation energy ranges from the command line
Cannot read the udem.inp file ...
taking the emission energy ranges from the command line
Cannot read the gfsp.inp file ...
No Ground-Final State Pairs will be evaluated
-------------------------------------------------------------------------------
PLOTTING RIXS SPECTRA
-------------------------------------------------------------------------------
Input File : test.el_inel.rocis.rixssoc
Incident Energy Excitation axis :   871.000 ...   876.000 eV  125 points
Energy transfer axis            :    -1.000 ...     4.000 eV  125 points
Incident Energy Shift         :    20.000 eV
Lorenzian Linewidth along Incident Axis                  :     0.400 eV
Lorenzian Linewidth along Energy Transfer/Emission Axis   :     0.400 eV
y axis  :     1 -> Energy transfer
Number of user defined cuts at constant Excitation Energy axis:    0
Number of user defined cuts at constant Emission/Energy Transfer Energy axis  : 0
```

(continues on next page)

(continued from previous page)

```
Making checks...Done

Proccessing data...
10% done
20% done
...
100% done

RIXS-plotting done
Incident Energy range:            845.800 ...   869.249
Emission/Energy-transfer range:     0.000 ...     4.853

Now storing the 2D file...
Done

Making the Integrated spectra along Energy Transfer/Emission axis... Done

Making the Integrated spectra along Incident axis... Done

All Done
--------------------------------------------------------------------------------
```

Successful run will generate the following files: The RIXS planes of the Total, Direct and Interference RIXS intensity as indicated in the above equations:

```
test.el_inel.rocis.rixssoc.total_rixs.dat
test.el_inel.rocis.rixssoc.direct_rixs.dat
test.el_inel.rocis.rixssoc.interference_rixs.dat
```

In addition one obtains the integrated spectra at constant Incident energies (CIE):

```
test.el_inel.rocis.rixssoc.dw.dat
```

as well as at constant Emission/Energy Transfer energies (CEE/CET):

```
test.el_inel.rocis.rixssoc.wex.dat
```

Fig. 5.25: DFT/ROCIS calculated RIXS planes for $[Cu(NH_3)_4]^{2-}$. Left: Total RIXS Intensity, Middle: Direct RIXS intensity and Right: Interference RIXS intensity. Lorentzian lineshape broadening with constant widths along Incident and Energy Transfer axis (0.5 and 0.2 eV respectively) were used throughout.

Generating Cuts

Cuts along x and y axis can be generated with two ways:

1) At first, this action can be performed by adding the following keywords: `uex` and `udw` accounting for generating cuts at constant Incident Energies (CIE) and at constant Emission (CEE)/or at constant Energy Transfer (CET) respectively, together with the desired number of cuts.

2) Alternatively, the energies of the desired cuts can be specified as lists in the files udex.inp (user defined excitations) udem.inp (user defined emissions)

For example if in udex.inp one specifies:

```
872.5
874.2
```

and for the cuts along Energy Transfer axis one just specify -udw3

```
orca_mapspc test.el_inel.rocis.rixssoc RIXS -x0871 -x1876 -x2-1 -x34 -w0.4 -g0.4
        -l -n125 -m125 -dx20 -eaxis1 -udw3
```

Then at the end one gets:

```
Making the specified  cuts (2) at constant Excitation Energy axis...
        Writing file: test.el_inel.rocis.rixssoc_872.50.rxes_vs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_872.50.rxes_fs.dat  ...Done
```

(continues on next page)

(continued from previous page)

```
        Writing file: test.el_inel.rocis.rixssoc_874.20.rxes_vs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_874.20.rxes_fs.dat  ...Done
        Done

        Making the specified cuts (3) at constant Emission/Energy Transfer axis...
        Writing file: test.el_inel.rocis.rixssoc_-1.00.xas_vs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_-1.00.xas_fs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_1.50.xas_vs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_1.50.xas_fs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_4.00.xas_vs.dat  ...Done
        Writing file: test.el_inel.rocis.rixssoc_4.00.xas_fs.dat  ...Done
        Done
        All Done
        ---------------------------------------------------------------------------
↪------
```

The files `*_rxes_fs.dat` are RXES spectra containing all individual contributions from all Final states together with the Direct, the Total and the Interference contributions at the given constant Incident Energy.

Similarly, the `*_rxes_vs.dat` are RXES spectra containing individual contributions of the Intermediate states, together with the Direct the Total and the Interference contributions at the given constant Incident Energy

Likewise, the respective `*_xas_fs.dat` and `*_xas_vs.dat` are XAS type spectra with individual contributions at a given constant Emission or Energy transfer Energy

These files are Energy vs Intensity files and read like:

1) for `*fs.dat`

```
X     S-  1(  0- 0)      S-  2(  0- 1)       DIRECT       TOT       INTERFERENCE
```

2) for `*vs.dat`

```
X     S-  1(  45)        S-  2(  47)         DIRECT       TOT       INTERFERENCE
```

In the first case S -1(0-0) represents the individual contribution of a given Ground-Final state pair. The numbering follows the numbering of the output file e.g.:

```
Eigenvalues:     cm-1         eV      Boltzmann populations at T =  300.000 K
        0:          0.0000     0.0000         3.44e-01
        1:          8.3818     0.0010         3.31e-01
```

Hence, in this case S -1 represents the elastic scattering intensity.

In the second case S -1(45) represents the individual contribution of a given Intermediate state.

```
44:      66918.6071     8.2968         1.43e-140
45:    6996678.8061   867.4775         0.00e+00
46:    6996693.0276   867.4793         0.00e+00
```

In this case S -1 represents the intensity contribution of the first Intermediate state.

Starting from ORCA 4.2 in every RIXS requested calculation the Off resonant XES spectrum is automatically generated in every RIXS requested calculation.

```
-------------------------------------------------------------------------------
ROCIS RIXS SPECTRUM
-------------------------------------------------------------------------------

Making the RIXS data files for Inelastic and Elastic Scattering
Ground State:                  1
Intermediate States:           28
Final States:                  588
```

(continues on next page)

(continued from previous page)

```
The RIXS cross section will be generated from:
589 Ground-Final State Pairs and   28 Intermediate States/Pair
The Off-Resonance XES spectrum will be printed

Calculating Intensities...
10% done
20% done
30% done
40% done
50% done
60% done
70% done
80% done
90% done
100% done

Printing the XES spectrum and Storing the files...
-------------------------------------------------------------------------------
 ↪--
X-RAY EMISSION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
 ↪--
Transition          Energy           INT             TX        TY        TZ
(eV)            (fosc)          (au)      (au)      (au)
-------------------------------------------------------------------------------
 ↪--
1  589 ->    0      6403.377     0.000000000721     0.00000   0.00000   0.00000
2  590 ->    0      6403.380     0.000000000083    -0.00000   0.00000   0.00000
3  591 ->    0      6403.685     0.000873238810     0.00236   0.00000   0.00000
4  592 ->    0      6404.766     0.000000000154     0.00000   0.00000   0.00000
5  593 ->    0      6408.288     0.000000006850    -0.00001   0.00000   0.00000
6  594 ->    0      6408.295     0.000034710300    -0.00047   0.00000   0.00000
...
16490  614 ->  588      6387.989     0.000000000000     0.00000   0.00000   0.00000
16491  615 ->  588      6388.222     0.000000000000     0.00000   0.00000   0.00000
16492  616 ->  588      6388.881     0.000000000000     0.00000   0.00000   0.00000
All Done
-------------------------------------------------------------------------------
```

Hence also the myfile-rixs.out file can also be processed with the `orca_mapspc` to generate the respective XES spectra:

```
orca_mapspc myfile_rixs.out XES/XESSOC -x06000 -x16500 -w2.0 -eV -n10000
```

5.7.5 ROCIS Magnetic Properties

Several magnetic properies are availiable in the ROCIS method Including g-tensors (G-Matrix), zero field splittings (ZFS), hyperfine couplings (HFCs) and electric field gradients (EFGs).

The g-tensors as well as the zfs are calculated on the basis of the Effective Hamiltonian as well in the sum over states (SOS) framework. HFCs are calculated in the SOS framework while EFGs are calculated as expectation values. Please consult also the respective discussion in the MRCI chapter (section *Multireference Configuration Interaction and Pertubation Theory (uncontracted)*)

```
...
DoHeff true     # Requests calculation of G-tenosrs and ZFS
                # in the effective Hamiltonian framework
DoEPR true      # Requests calculation of G-tenosrs, ZFS and HFCs
                # in the Sum over states (SOS) framework
```

(continues on next page)

(continued from previous page)

```
AtensorNuc 0   # Nuclei to account for the HFCs calculation
NAtensors 1    # How many Nuclei are included in the HFCs calculation
ATensor 0      # Nucleus to calculate HFCs and EFGs
NDoubGtensor 1 # Kramers doublets to account for the g tensor calculations
...
```

This will enter the calculation in the ROCIS Spin Hamiltonian section

```
-------------------------------------------------------
ROCIS SPIN HAMILTONIAN PROPERTIES
-------------------------------------------------------
```

5.7.6 Natural Transition Orbitals/ Natural Difference Orbitals

Likewise to CIS and TD-DFT (section *Natural Transition Orbitals*) The nature of the calculated excited states in ROCIS and DFT/ROCIS can be analyzed by using the Natural Transition Orbitals (NTO) or Natural Difference Orbitals (NDO) machineries.[658]

Note

- The NTO analysis is based on the transition density between ground and excited states. Hence is valid for singly excited states and for states of the same multiplicity.
- The NDO analysis on the otherhand is somewhat more flexible in this respect as it is based on the difference density between ground and excited states.
- Presently, only one analysis (NTO or NDO) can be performed at a time while when both flags are on the NTO analysis switches off.

To request for NTOs or NDOs the user needs to provide which states are to be calculated:

```
DoNTO true
NTOStates = 1,2,3,4,5,6,7,8,9,10,13,14,15
```

```
DoNDO true
NDOStates = 1,2,3,4,5,6,7,8,9,10,13,14,15
```

Which gives:

```
-----------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE   14
-----------------------------------------

done
Solving eigenvalue problem for the Occupied space  ... done
Solving eigenvalue problem for the Acceptor space   ... done
Natural Transition Orbitals were saved in nto.14.nto
Threshold for printing occupation numbers 1.0e-04

E=  25.447756 au    692.469 eV  5585137.0 cm**-1
49[0] ->  46[1]  : n=  0.39056909
48[0] ->  47[1]  : n=  0.08619374
47[0] ->  48[1]  : n=  0.00441125
```

```
-------------------------------------------------
NATURAL DIFFERENCE ORBITALS FOR STATE   14
-------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
done
Solving eigenvalue problem for the Occupied space  ... done
Solving eigenvalue problem for the Acceptor space   ... done
Natural Difference Orbitals were saved in ndo.14.ndo
Threshold for printing occupation numbers 1.0e-04

E=  25.447756 au    692.469 eV  5585137.0 cm**-1
49[0] ->  46[1]  : n=  0.81173217
48[0] ->  47[1]  : n=  0.17903699
47[0] ->  48[1]  : n=  0.01165859
46[0] ->  49[1]  : n=  0.00922738
45[0] ->  50[1]  : n=  0.00112567
```

For closed shell cases the orbitals are save in similar way to TDDFT and CIS (section *Natural Transition Orbitals*). In the case of open shell cases for convenience donor orbitals are saved with orbital operator 0 while acceptor orbitals with orbital operator 1. This needs to be specified in the `orca_plot` program and should not be confused with the `spin-up` and `spin-down` orbitals in the UHF and UKS cases.

5.7.7 Keyword List

Table 5.9: `%rocis` block input keywords.

Keyword	Option / Value	Description
`DoGenROCIS`	`true`	Requests the GS-ROCIS method (default: `false`).
`ReferenceMult`	`1`	Sets the multiplicity for the reference CSF for the GS-ROCIS method.
`NRoots`	`5`	The number of desired roots.
`MaxDim`	`25`	Davidson expansion space = MaxDim * NRoots.
`MaxIter`	`100`	Maximum CI Iterations.
`NGuessMat`	`512`	The dimension of the guess CI matrix.
`ETol`	`1e-6`	Energy convergence tolerance.
`RTol`	`1e-6`	Residual Convergence tolerance.
`MaxCore`	`4096`	Maximum memory used by the ROCIS moduleduring the calculation in MB.
`EWin =`	`-5,5-5,5`	Energy Window that defines orbital excitation space.
`OrbWin =`	`6,8,0,2000`	Orbital Window that defines orbital excitation space (overrides EWin).
`DoLoc`	`true`	Switch for localization of Donor orbital space (default: `false`).
`LocOrbWin =`	`6,11`	Obital window for the localization scheme.
`LocMet`	`PipekMezey` PM	Chooses the Pipek Mezey localization method.
	`FosterBoys` FB	Chooses the Foster Boys localization method.
`DoPNO`	`true`	Performs the calculation in the PNO-ROCIS framework (default: `false`).
`TCutPNO`	`1e-16`	Threshold for PNO selection.
`XASelems`	`0,2`	Selects from which atoms are the core MOs going to be used in the PNO procedure.
`LocMet`	`PipekMezey` PM	Chooses the Pipek Mezey localization method.
	`FosterBoys` FB	Chooses the Foster Boys localization method.
`DoNTO`	`true`	Request Natural Transition Orbital Analysis (default: `false`)
`NTOThresh`	`1e-4`	Threshold for printing occupation numbers.
`NTOStates =`	`1,2`	Array input for states to be taken into account.
`DoNDO`	`true`	Request Natural Difference Orbital Analysis (default: `false`)
		(if true it switches off the NTO analysis)
`NDOThresh`	`1e-4`	Threshold for printing occupation numbers.
`NDOStates =`	`1,2`	Array input for states to be taken into account.
`PrintLevel`	`3`	Controls the amount of output produced during the calculation.
`Do_is`	`true`	Include DOMO->SOMO excitations.
`Do_sa`	`true`	Include SOMO->Virtual excitation.
`Do_ia`	`true`	Include DOMO->Virtual excitations.
`Do_ista`	`true`	Include DOMO->SOMO excitations coupled to SOMO->Virtual excitations with s$\neq$ t.
`Do_isa`	`true`	Include DOMO->SOMO excitations coupled to SOMO->Virtual excitations with s= t.
		(only used in the original ROCIS implementation)
`DoLowerMult`	`true`	Switch for excitation with S'=S-1.
`Do_LM_is`	`true`	Include DOMO->SOMO excitations with S'=S-1.
`Do_LM_sa`	`true`	Include SOMO->Virtual excitations with S'=S-1.
`Do_LM_ia`	`true`	Include DOMO->Virtual excitations with S'=S-1.
`Do_LM_ss`	`true`	Include SOMO->SOMO excitations with S'=S-1.
`DoHigherMult`	`true`	Switch for excitations with S'=S+1.
`Do_HM_is`	`true`	Include DOMO->SOMO excitations with S'=S+1.

continues on next page

Table 5.9 – continued from previous page

Keyword	Option / Value	Description
Do_HM_sa	true	Include SOMO->Virtual excitations with S'=S+1.
Do_HM_ia	true	Include DOMO->Virtual excitations with S'=S+1.
DoCD	true	Request circular dichroism calculation.
DoDipoleLength	true	Request the use of electric moments in a length formulation (default: false).
DoDipoleVelocity	true	Request the use of electric moments in a velocity formulation (default: false).
DoFullSemiclassical	true	Request the calculation of complete semiclassical multipolar moments (default: false).
DecomposeFosc	true	Request the decomposition of the oscillator strengths in a multipolar expansion. (default: false)
RIXS	true	Requests a RIXS calculation (default: false).
RIXSSOC	true	Perform a RIXS calculation on the basis of relativistically corrected states (default: false).
Elastic	true	Include the elastic line in the generation of the RIXS or RIXSSOC spectra (default: false).
PlotDiffDens =	1,2	Array input for plotting difference densities of CI roots to the ground state.
PlotSOCDiffDens =	1,2	Array input for plotting difference densities of SOC states to the ground state.
TPrint	0.01	Threshold for contributions to CI and SOC states to be printed.
rel		Signals the begining of the relativistic block.
DoSOC	true	Requests the inclusion of spin-orbit coupling (part of the rel block).
DoMCD	true	Requests the calculation of MCD spectra (part of the rel block).
B	30000	Sets external magnetic field (part of the rel block).
Temperature	5	Sets the temperature for the SOC spectra (part of the rel block).

5.8 Excited States via MCRPA

The MCRPA method is a synonym for *CASSCF* linear response theory. Having converged *CASSCF* calculations for the electronic ground state, you can employ ORCA's MCRPA implementation to compute

- excitation energies
- oscillator strengths (intensities of the UV/Vis spectrum)
- rotatory strengths (intensities of the ECD spectrum)
- core excitation energies to simulate XAS spectra primarily for open-shell and strongly correlated systems.

Before running these calculations make sure you are familiar with *CASSCF* calculations in general.

5.8.1 General Description

MCRPA excitation energies and transition moments are computed from the poles and residues of the linear response function of CASSCF a wave function.[592, 593, 594] Those are determined from the generalized eigenvalue equations

$$\mathbf{E}^{(2)}\,\boldsymbol{\lambda} = \mathbf{S}^{(2)}\,\boldsymbol{\lambda}\,\omega,$$

that involve the second-derivative matrices $\mathbf{E}^{(2)}$ and $\mathbf{S}^{(2)}$. The eigenvalues ω correspond to the electronic excitation energies. The second-derivative matrices $\mathbf{E}^{(2)}$ and $\mathbf{S}^{(2)}$ have a paired structure as both kind of operators that express electronic excitation and de-excitations are involved:

$$\left[\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^* & \mathbf{A}^* \end{pmatrix} - \omega \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Delta} \\ -\boldsymbol{\Delta}^* & -\boldsymbol{\Sigma}^* \end{pmatrix}\right] \begin{pmatrix} \mathbf{X} \\ \mathbf{Y}^* \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

The eigenvalue equations above are valid for all variational wave functions methods, e.g. *DFT*, *HF*, *CASSCF* etc. The only difference is the

operator manifold and the time-independent Hamiltonian $\hat{H}$ that is used. For the CASSCF linear response and eigen value equations, the super matrices $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\Delta}$ have the following structure:

$$\mathbf{A} = \begin{pmatrix} \langle 0|\,[q_i,[\hat{H},q_j^\dagger]]\,|0\rangle & \langle 0|\,[[q_i,\hat{H}],R_j^\dagger]\,|0\rangle \\ \langle 0|\,[R_i,[\hat{H},q_j^\dagger]]\,|0\rangle & \langle 0|\,[R_i,[\hat{H},R_j^\dagger]]\,|0\rangle \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \langle 0|\,[q_i,q_j^\dagger]\,|0\rangle & \langle 0|\,[q_i,R_j^\dagger]\,|0\rangle \\ \langle 0|\,[R_i,q_j^\dagger]\,|0\rangle & \langle 0|\,[R_i,R_j^\dagger]\,|0\rangle \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} \langle 0|\,[q_i,[\hat{H},q_j]]\,|0\rangle & \langle 0|\,[[q_i,\hat{H}],R_j]\,|0\rangle \\ \langle 0|\,[R_i,[\hat{H},q_j]]\,|0\rangle & \langle 0|\,[R_i,[\hat{H},R_j]]\,|0\rangle \end{pmatrix} \quad \boldsymbol{\Delta} = \begin{pmatrix} \langle 0|\,[q_i,q_j]\,|0\rangle & \langle 0|\,[q_i,R_j]\,|0\rangle \\ \langle 0|\,[R_i,q_j]\,|0\rangle & \langle 0|\,[R_i,R_j]\,|0\rangle \end{pmatrix}$$

There are two types of operators for describing electronic excitations, i.e. orbital excitation $q_i^\dagger$ and de-excitation operators q_i,

$$q_i^\dagger = E_{pq} = a_{p\alpha}^\dagger a_{q\alpha} + a_{p\beta}^\dagger a_{q\beta}, \quad q_i = E_{qp}, \quad \text{with} \quad p > q$$

as well as so called state-transfer operators

$$R_i^\dagger = |i\rangle\langle 0|, \quad R_i = |0\rangle\langle i|, \quad \text{with } i \neq 0$$

Intensities of the (UV/Vis) absorption and electronic circular dichroism (ECD) spectra are obtained from the transition moments. Transition moments are easily computed as dot product

$$\langle 0|\hat{V}|f\rangle = \begin{pmatrix} \mathbf{X}^f \\ (\mathbf{Y}^f)^* \end{pmatrix}^\dagger \begin{pmatrix} \mathbf{C} \\ \mathbf{D}^* \end{pmatrix}$$

from the eigenvectors **X** and **Y** of root f and the electronic gradients **C** and **D** of the dipole operators $\hat{V}$.

$$\mathbf{C} = \begin{pmatrix} \langle 0|[q_i, \hat{V}]|0\rangle \\ \langle 0|[R_i, \hat{V}]|0\rangle \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} \langle 0|[q_i^\dagger, \hat{V}]|0\rangle \\ \langle 0|[R_i^\dagger, \hat{V}]|0\rangle \end{pmatrix}$$

The eigenvalue equations are solved iteratively by a customized version of the Davidson algorithm that simultaneously determines the N lowest lying roots. The most time of the calculation is spent on the transformation of trial vectors that contain an orbital and CI coefficient part with the electronic Hessian matrix $\mathbf{E}^{(2)}$. The working equations are very similar to those of the CASSCF electronic gradient that is computed when minimizing the CASSCF ground state energy. Further technical aspects are described in subsection *Computational Aspects*.

Fig. 5.26: Structure and frontier orbitals of Ni(II) hexaquo complex.

As a show case example, the lowest valence excitation energies of the octahedrally coordinated Ni(II) hexaquo complex are computed with MCRPA (for structure and frontier orbitals see Fig. Fig. 5.26). We include the 3d and 4d Ni orbitals in the active space and converge the CASSCF triplet ground state, first. Then, the MCRPA calculation is launched. It is only necessary to specify the number of desired roots `NRoots`.

```
%casscf
 nel    10  # number of active electrons
 norb   10  # number of active orbitals
 nroots  1  # only the ground state
 mult    3  # triplet multiplicity
end

%mcrpa
 NRoots 15  # number of roots / excited states
end
```

Note that in contrast to state-averaged CASSCF, the MCRPA excitation energies do not dependent on `NRoots` as for *TDDFT* or *EOM-CC* calculations.

It is also possible to omit the orbital part of MCTDA or MCRPA calculations by toggling the `DoOrbResp false` flag. For such calculation, the *CASCI* excitation energies are obtained. Note that for such MCRPA calculations the orbitals were optimized for the CASSCF ground state.

5.8.2 Analyzing Transitions

After convergence, the most significant contributions of the excitation / eigen vector is given for each state. When comparing with the ground state CI coefficients

```
---------------------------------------------
CAS-SCF STATES FOR BLOCK  0 MULT= 3 NROOTS= 1
---------------------------------------------

ROOT   0:  E=   -1962.8698444660 Eh
      0.98088 [     0]: 2221100000
```

we see that the lowest excited state is dominated by a transition from active orbital 3 (dyz) to active orbitals 4 and 5 (mixture of dz2 and dx2-y2).

```
STATE    1 3A  :  E=   0.033027 au      0.899 eV     7248.5 cm**-1


   (+) CI                                  0.72808 [    21]: 2212100000
   (+) CI                                  0.25245 [    27]: 2211200000
```

The `(+)` denotes the upper component of the eigenvectors.

By default the largest elements of the CI eigenvector are printed in the configuration basis. By changing `PrintWF` to `DET` or `CSF` you can also print that in the determinant or configuration state function (CSF) basis, respectively.

Note that the analysis of CI coefficient depends also on the choice of active orbitals which are chosen to be natural orbitals (default option).

There are also transitions dominated by single-electron excitations represented by the orbital part of the eigenvector. In case of the Ni(II) hexaquo complex those are primarily Rydberg states. Below find the most significant orbital pairs (upper `(+)` and lower `(-)` component) for the 24th excited state.

```
STATE   24 3A  :  E=   0.469596 au     12.778 eV   103064.3 cm**-1

   (+) I  24 (  35) -> V   0 (  49) :      0.01744 (c=  0.13206324)
   (+) I  26 (  37) -> V   2 (  51) :      0.02398 (c= -0.15485895)
   (+) I  24 (  35) -> V   4 (  53) :      0.02513 (c=  0.15853501)
   (+) A   0 (  39) -> V   0 (  49) :      0.26647 (c= -0.51620412)
   (+) A   7 (  46) -> V   0 (  49) :      1.55291 (c=  1.24615978)
   (+) A   0 (  39) -> V  16 (  65) :      0.11782 (c= -0.34324922)
   (+) A   7 (  46) -> V  16 (  65) :      0.73241 (c=  0.85580859)


   (-) I   0 (  11) -> A   0 (  39) :      0.05738 (c=  0.23954155)
```

5.8.3 Natural Transition Orbitals

Rather than analyzing the sections of the excitation vectors directly, visualizing natural transition orbitals[594, 595] (NTO) can be very insightful. Since NTOs are obtained from a singular value decomposition of the MCRPA ground-to-excited state (f) transition density matrices $\rho_{pq}^{0\to f}$, they are independent of any unitary transformation of the molecular orbital (inactive, active, and virtual) sections and, thus, independent of type of orbitals (canonical, localized, natural, etc.).

As for *TDDFT* and *ROCIS* one obtains two sets of orbitals for each state that describe the donation (occupied and active) and acceptance (active and virtual) of an electron in the electronic transition. The orbital structure of $\rho_{pq}^{0\to f}$ for CASSCF wave functions is illustrated in Fig. 5.27.

Fig. 5.27: Structure of MCRPA transition density matrix $\rho_{pq}^{0\to f}$

Note that the configuration part of the excitation vector contributes to the active-active block of $\rho_{pq}^{0\to f}$ and is thus respected.

To compute NTOs, only the `DoNTO` flag in the input has to switched on (`DoNTO true`):

```
%mcrpa
 nroots    20    # number of roots
 DoNTO     true  # generate NTOs for converged roots
 NTOThresh 5e-5  # NTO selection threshold
end
```

This will compute all NTOs with a singular value larger then the `NTOThresh` threshold for ALL roots.

```
-----------------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE     1 3A
-----------------------------------------------

STATE    1 3A  :  E=   0.033027 au      0.899 eV     7248.5 cm**-1

Threshold for printing occupation numbers 1.0000e-03

     0  : n=  0.99567919
     1  : n=  0.02731085
     2  : n=  0.02231787
     ...

=> Natural Transition Orbitals (donor   ) were saved in ni-II-h2o-6-cas-8-10-mcrpa.
↪mcrpa.1-3A_nto-donor.gbw
=> Natural Transition Orbitals (acceptor) were saved in ni-II-h2o-6-cas-8-10-mcrpa.
↪mcrpa.1-3A_nto-acceptor.gbw
```

For the above example, the most important (controlled by `NTOThresh`) donating and accepting NTOs of state 1 are written to the `gbw`-type files. The NTOs can be visualized with the `orca_plot` program (see Sec. *orca_plot*), for the donating NTOs

```
orca_plot ni-II-h2o-6-cas-8-10-mcrpa.mcrpa.1-3A_nto-donor.gbw -i
```

and for the accepting NTOs

```
orca_plot ni-II-h2o-6-cas-8-10-mcrpa.mcrpa.1-3A_nto-acceptor.gbw -i
```

The NTOs are ordered according to their singular value in descending order as given in the output, starting with index 0 for the most significant NTO pair.

The most dominant NTO pairs (index 0) for the three lowest valence 3Tg transitions of Ni(H2O)62+ are given in Fig. Fig. 5.28.

Fig. 5.28: Lowest transitions in Ni(H2O)62+.

5.8.4 X-Ray Absorption Spectroscopy

It is also possible to compute X-ray absorption spectra (XAS) with the MCRPA module by making use of the core-valence separation (CVS) approximation.[659] For such calculations, the same active space as for ground state CASSCF or valence excitation energy MCRPA calculations should be chosen. To access MCRPA core excitations, two restrictions are introduced by the CVS approximation:

- The configuration part of the eigenvectors is omitted as it is responsible for valence excitations within the valence active orbitals.
- The orbital part of the eigenvectors is restricted to core-active and core-virtual pairs with a user given list of core orbitals

For computing the lowest MCRPA oxygen K-edge transitions of ozone, one can employ the following input for an CVS-MCRPA calculation:

```
%casscf
 nel    12 # number of active electrons
 norb    9 # number of active orbitals
 nroots  1 # only the ground state
 mult    1 # singlet multiplicity
end

%mcrpa
 NRoots 15            # number of roots / excited states
```

(continues on next page)

(continued from previous page)

```
 DoCVS  true          # turn on core valence separation
 OrbWin 0, 2, -1, -1  # select on the lowest 3 core orbitals
 DoNTO  true          # generate NTOs for converged roots
end
```

Here the first 3 inactive MOs are selected by `OrbWin` that are primarily expanded in the 3 1s atomic orbitals of the three O atoms. There is no restriction on the virtual orbitals (`-1` is chosen for first and last virtual MO index).

Note that in constrast to other *multi-reference protocols*, the core orbitals must not get added to the active space for CVS-MCRPA calculations.

The most dominant NTO pairs for the four lowest core excitations are given below:

Fig. 5.29: Lowest CVS-MCRPA oxygen K-edge transitions in ozone.

5.8.5 Computational Aspects

Large Molecules

The code is intended to be used for medium-sized and larger open-shell molecules. It has the same scaling as ORCA's first-order CASSCF energy implementation though a larger pre-factor as the computational cost grow "in principle" linearly with the number of roots.

The implementation is AO-driven meaning that the computational bottleneck is the Fock matrix construction for the several state-specific pseudo AO densities. Note that there are up to 6 pseudo AO densities for each state. The computational costs can be reduced significantly if the *RIJCOSX* approximation is employed, which is highly recommended.

The second most expensive part of the MCRPA computation are the two-electron integrals with 3 active indices g_{ptuv}. As we aim for running calculations on larger systems, there is only an implementation of the integral transformation that uses the resolution-of-the-identity (RI) approximation.

The restrictions on the auxiliary basis sets are the same as for the CASSCF code (Sec. *CASSCF Linear Response*). That is

- If the Fock matrices are constructed in `Direct` or `Conventional` mode, the /C bases are used for the RI approximation of the g_{ptuv} integrals.
- If the *RIJCOSX* approximation for the Fock matrices is employed, the /JK bases are used for both the Fock matrices and the g_{ptuv} integrals.

Note that the *RIJK* approximation is not yet available.

Convergence

Before starting running MCRPA, it is recommended to converge the state-specific CASSCF energy calculation very tightly. The *TRAH-CASSCF* is ideally suited for that purpose. Note that property calculations in general assume vanishing electronic gradients otherwise numerical issues in the eigenvalue / response equations may occur.

Eigenvalues and eigenvectors are determined by a dedicated variant of the Davidson algorithm for solving large-scale TDA or RPA eigenvalue equations.[594, 660] The accuracy of the eigenvalues and eigenvectors is solely determined by `TolR` (default 1.e-5) that is compared to the Frobenius of the residual vector for each root.

Similar to converging CASSCF equations, the MCTDA and MCRPA convergence rates are also affected by the choice of molecules orbitals. This has to be setup in the `%casscf` input block. The MCRPA program will not change the converged orbitals and CI coefficients. For most calculations, the default settings are most appropriate which are *canonical orbitals* for inactive and virtual orbitals and *natural orbitals* for active orbitals. as explained in subsection *Final Orbitals: Canonicalization Choices*. For multi-center systems as bimetallic transition metal complexes, *localized orbitals* are a better choice. Purified d or f orbitals (`ActOrbs dorbs` or `ddorbs` and `forbs`) for single-metal complexes are often helpful for analyzing excitations, though, their choice usually result in slower convergence of the eigenvalue equations.

To facilitate quick convergence of the eigenvalue equations, we start the iterations from eigenvectors that were obtained from direct diagonalization of full Hessian and metric matrices that were compute for a subset of elements. This is referred to as P-space or `full` method which is the default for obtaining initial eigenvectors but also the preconditioner of the TDA/RPA Davidson algorithm (`PreCond full`). The P-space method contains the orbital-orbital and configuration-configuration blocks that are diagonalized individually. Orbital-configuration coupling is omitted here. The size of the P space is controlled by the keyword `PreConMaxRed` which is set to 400 by default. Faster convergence can be expected with larger P-space dimensions, though the cost of the full diagonalization increase cubically. Alternatively, the approximate diagonal Hessian[661] and metric can be used for preconditioning (`PreCond diag`), but not for the initial eigenvectors. It is recommended to employ `Precond full`.

5.8.6 Keyword List

```
%mcrpa

  NRoots       0         # The number of desired roots

  TolR         1e-5      # Convergence threshold for residual norm
  PrintWF      CFG       # print CI part in (CFG (default), CSF, or DET basis)
  TolPrint     1e-2      # Threshold for printing elements of excitation vector

  MaxRed       200       # maximum size of reduces space for ALL VECTORS IN TOTAL
  MaxIter      100       # maximum number of (Davidson) iterations

  TDA          false     # Switch off full TD-CASSCF (Tamm-Dancoff approximation)

  DoNTO        false     # Generate Natural Transition Orbitals
  NTOThresh    1e-3      # Threshold for printing occupation numbers

  DoCVS        false     # toggle core valence separation (CVS) for XAS
  OrbWin       i0, i1, a0, a1 # first and last inactive and virtual MO,␣
↪respectively

  PreCond      full      # (default) P-space preconditioner uses parts of exact␣
↪Hessian matrix
               diag      #           diagonal preconditioner approximate diagonal␣
↪Hessian elements

  PreConMaxRed 400       # dimension of exact Hessian for preconditioning

  DoOrbResp    true      # turn orbital response off -> CAS-CI method

  DoCD                  true  # circular dichroism calculation
  DoDipoleLength        true  # electric moments in a length formulation
  DoDipoleVelocity      true  # electric moments in a velocity formulation
  DoFullSemiclassical   false # request the calculation of complete semiclassical␣
↪multipolar moments
  DecomposeFosc         false # request the decomposition of the oscillator␣
↪strengths in a multipolar expansion.
```

5.9 Excited States via EOM-CCSD

Many of the previously described methods are based on the single excitation framework. For a more accurate treatment, double excitations should also be considered. The equation of motion (EOM) CCSD method (and the closely related family of linear response CC methods) provides an accurate way of describing excited, ionized, and electron-attached states based on singles and doubles excitations within the coupled-cluster framework.

5.9.1 Theory

The EOM wave function is parametrized in the following manner

$$\mathcal{R}|\Psi_{CC}\rangle,$$

i.e. via the action of a linear excitation operator $\mathcal{R}$ on the coupled-cluster ground state wave function Ψ_{CC}. Here, $\mathcal{R}$ is a particle conserving operator, in the case of the excitation energy problem. However, this is not true for the ionization or electron attachment problems, where $\mathcal{R}$ is a net annihilation or net creation operator, respectively. The ground state coupled-cluster T-amplitudes are obtained from a CCSD calculation, and our task is to obtain $\mathcal{R}$. Note that since the CC Hamiltonian is nonsymmetric, a left hand solution ($\mathcal{L}$) would also be needed to evaluate properties. For the calculation of excitation, ionization or electron attachment energies, however, it is enough to obtain the right

hand solutions ($\mathcal{R}$). In principle, this is done by building the Hamiltonian and diagonalizing it in order to obtain energy expectation values.

In practice, the size of the CCSD Hamiltonian matrix is prohibitively large and thus, various methods have been devised to obtain its lowest few eigenvalues and eigenstates. One of the most popular of these approaches is the Davidson method, which can be summarized as follows:

- Construct an initial guess of orthogonal trial vectors, C.
- Evaluate the sigma vectors $\sigma = HC$.
- Build model Hamiltonian $\mathcal{H} = C^T\sigma$.
- Diagonalize $\mathcal{H}$: $\mathcal{E} = \mathcal{U}^T\mathcal{H}\mathcal{U}$.
- Compute Ritz vectors $X = C\mathcal{U}$.
- Compute residuals $R = X\mathcal{E} - \sigma\mathcal{U}$, check convergence: if yes, pass $X, \mathcal{E}$ as solutions.
- Preconditioning: $T = MR$ (many possible choices for the preconditioner M).
- Check if adding new trial vectors would exceed the maximum number of trial vectors:
  - if no, add T to C, and orthonormalize the united set
  - if yes, then set X as C (orthonormalize if H is nonsymmetric); then add T and orthonormalize

The advantage of the above method is that, instead of the full Hamiltonian, only the sigma vectors have to be explicitly evaluated and stored.

It is also possible to use a lower scaling version of the EOM-CCSD methods, which relies on the perturbative truncation of the coupled-cluster similarity transformed Hamiltonian. Presently, only the second order truncated version (CCSD(2) approximation) is available for closed-shell molecules (RHF). However, it is better to use the PNO based implementation, as it has the cost of EOM-CCSD(2), but its accuracy is comparable to canonical EOM-CCSD.

Initial Guess

The present initial guess in the RHF EOM implementation consists of constructing a CIS Hamiltonian of a certain dimension, and diagonalizing it. The roots are preselected based on the energetic ordering of the diagonal elements of the Hamiltonian. In the UHF case, the guess is constructed from the solutions of a UHF CIS calculation. The number of roots in the initial guess is determined as 20 times the number of roots desired in EOM (`NRoots`) if `NDav` is 20 or smaller, otherwise it is set to `NDav` times the number of EOM roots. If the parameter `NInits` is larger than zero, then the number of initial guess roots will be set to this parameter times `NRoots`. The maximum possible number of roots is the full CIS dimension, ($N_O N_V$ (RHF) or $N_{O_\alpha} N_{V_\alpha} + N_{O_\beta} N_{V_\beta}$ (UHF)) . One should keep in mind, while increasing the number of initial guess vectors, that this corresponds to diagonalizing a matrix of increasing dimension. If, for example `NRoots` is 10, then by default 200 roots are considered in the initial guess (unless it exceeds the size of the CIS space), or if `NInits` is set to 100, then there will be 1000 roots in the guess. In some cases, the roots calculated using EOM may not be the lowest ones, but a few of these may be replaced by some higher roots which are "easier" to find. In such cases, it may help to increase `NRoots` or `NInits` to converge to the proper roots. The program can be made to follow the initial CIS guess by setting `FollowCIS` to true and is necessary if we wish to ionize or excite from inner-valence or core orbitals. In the RHF implementation, the core orbital, from which the ionization or excitation originates, can be specified using the keyword `CoreHole`, in addition to setting `DoCore` and `FollowCIS` to true. The `CoreHole` keyword is quite general and in principle, ionization or excitation processes from any occupied orbital can be specified using this keyword.

Hamiltonian Construction

The Hamiltonian construction begins by calling the sigma routines. In the case of the closed-shell code, the logical variables `UseEOMOptS` and `UseEOMOptD` choose the routines to be used in the evaluation of the singles and doubles sigma vectors, respectively. If true, the optimized sigma routine, using dressed integrals, will be used. This should not be changed, the option is there mainly for debugging purposes. If set to false, an automatically generated, and much slower serial code will be used instead. In the case of the open-shell UHF implementation, optimized sigma routines have been generated using the ORCA Automated Generator Environment (AGE) [361]. In each early iteration, N_R sigma vectors will be determined, except in the case of a restart, where the number of sigma vectors is $2N_R$. For further details on convergence, see *Convergence, Restart, Preconditioning and Subspace Expansion* below.

The most time consuming part of the sigma vector construction is the formation of the external exchange contribution, which can be influenced via the CC keyword `KCOpt`. Currently, there are three options that are compatible with the RHF EOM implementation: `KC_MO`,`KC_AOX` and `KC_AOBLAS` and `KC_AOX` is the only option available in the UHF EOM code. The external exchange term can be treated most efficiently using COSX, which in the closed-shell case, leads to average speed ups of 10x for the external exchange term and an overall speedup of 3x for the EOM calculation. This is accompanied by a drastic reduction of the storage cost[662]. The error introduced is below 1 meV, which is 200-fold less than the error bar of the method[662] itself. It is the default for `KCOpt KC_AOX` and `KC_AOBLAS` and can be controlled by the keyword `DOCOSXEOM`. The default grid settings for EOM are `GridX 1` and `IntAccX 2.68`.

Once the sigma vectors are available, they are multiplied with the trial vectors to yield the reduced space Hamiltonian. The Hamiltonian is built in a way that, in each iteration, only the new vector products are added to the "edge" of the old Hamiltonian, so that a full build is avoided. It should be clear that the parameter `NDav` plays an important role here, since it determines the maximum size of the Hamiltonian ($N_D N_R$), and also controls how much memory is needed for the trial and sigma vectors, as seen above. Since the choice of this parameter influences convergence properties, it will be discussed further in *Convergence, Restart, Preconditioning and Subspace Expansion*.

Solution of the (Nonsymmetric) Eigenproblem

Following the construction of the Hamiltonian, a nonsymmetric eigensolver is called. In this case, it is possible to have complex eigenvalues. In practice, this is rarely the case, and indicates a problem of some kind. A warning will be given if this happens, however, one may get away with this if it only happens in an isolated iteration step.

Once the eigenvectors are available, they are compared with those of the previous iteration, if root homing is turned on, i.e. if the `RootHoming` keyword is set to true. This means evaluating the overlap of the old and new eigenvectors, in order to keep track of the possible movement of the eigenvectors if root flipping occurs. If converged roots are removed from further iterations (see next section), it is important to keep track of changes in ordering, especially if a converged and a non-converged root is swapped. After diagonalization, the Ritz vectors and residuals can be evaluated.

Convergence, Restart, Preconditioning and Subspace Expansion

Convergence is signaled once a residual square norm based criteria is fulfilled. This criteria is determined by the `CheckEachRoot` keyword. If it is true (default), the convergence of the residual square norm of each root is checked separately. This is due to the fact that different roots converge at a different rate. Once a root is converged, no new trial vectors will be generated, belonging to that vector. This means that the EOM iterations will progressively become faster (until restart). Turning off the rootwise convergence check is possible, but not recommended. In this case, the maximum of all residual square norms is checked for convergence, and all iterations will take roughly the same amount of time since no vectors are removed in any iteration. However, this procedure can be numerically unstable, since the residuals of some roots might become very close to zero, and trying to generate new vectors, which are orthogonal to these, may lead to numerical disaster. In short, the recommended default is having both `CheckEachRoot` and `RootHoming` set to true. If `CheckEachRoot` is false, then `RootHoming` should also be set to false, as it may cause problems if `NDav` is too small. The convergence threshold of the residual in Davidson's method can be larger than that for the ground state CC residual threshold in order to obtain converged results. Namely, a value of `DTol` of 1e-5 is almost always enough to get well converged energies.

At this point it is worth discussing the role of the keyword `NDav`. This keyword determines at what point the Davidson algorithm should be restarted. If it is chosen too small, it may cause slow convergence. If this value is too large, this

may result in overwhelming demands on memory/disk space requirements. The default value (20) is chosen with the hope that no, or maybe one restart will be required. It should only be changed if computational resources demand it. However, the treatment of core ionization or core excitation processes often requires a large value of `NDav`. At restart, the Ritz vectors are copied as new trial vectors for all roots, which will then be orthonormalized, while new vectors will only be generated for the non-converged roots. This means that the step after the rebuilding of the expansion space will be 1-2 times as expensive as one of the initial steps.

New directions (trial vectors) are generated from the preconditioned residual vectors. If no preconditioning is applied (the preconditioner is taken to be a unit matrix), one falls back to the Lanczos algorithm, which is inferior to the Davidson algorithm. This happens if the keyword `DoLanczos` is true. This is not recommended, as the Lanczos algorithm converges several times slower than Davidson's, and is there for debugging mainly. The original Davidson preconditioner is the inverse of a diagonal matrix which contains the difference of the diagonal elements of the Hamiltonian and the current approximation to the eigenvalue belonging to the given root. Let us consider the closed-shell RHF implementation for simplicity. If R_{ia} and R_{ijab} are elements of the singles and doubles amplitudes, respectively, then the updated vectors (T_{ia}, T_{ijab}) have the form

$$T_{ia} = \frac{R_{ia}}{D_{ia} + \mathcal{E}_R}$$

for singles, and

$$T_{ijab} = \frac{R_{ia}}{D_{ijab} + \mathcal{E}_R}$$

for doubles. Here, D_{ia} and D_{ijab} are related to, and possibly approximations of, the respective diagonal Hamiltonian elements. The simplest approximation is just to construct these from diagonal Fock matrix elements (i.e. orbital energies) as $D_{ia} = \varepsilon_a - \varepsilon_i$ and $D_{ijab} = \varepsilon_a + \varepsilon_b - \varepsilon_i - \varepsilon_j$. A slightly better preconditioning can be obtained as follows. For singles, take the exact CIS diagonal elements, $D_{ia} = \varepsilon_a - \varepsilon_i + \overline{g}_{iiaa}$, where the last term is the respective antisymmetrized integral; and construct the doubles as $D_{ijab} = D_{ia} + D_{jb}$. This is the default, and can be changed back to the simple Fock matrix guess by setting `UseCISUpdate` to false.

Following the preconditioning step, the resulting vectors are orthogonalized to the previous set of trial vectors, and orthonormalized among themselves. Since, the trial vectors do not change once they are generated (unless a restart occurs), only the new elements of the overlap matrix need to be generated for the orthonormalization. The numerical threshold for the inversion (and other division steps) is controlled by the parameter `OTol`. Finally, the amount of printed information can be controlled via the `PrintLevel` keyword. If not given or equal to 2, only basic iteration information will be printed. If set to 3, detailed iteration information will be printed (recommended if timing results for individual steps are required), while 4 or higher triggers additional (and very verbose) information from other subroutines as well.

The default solver is a multi-root Davidson procedure. The single-root solver can be initiated by setting `DoRootwise` and `FollowCIS` to true. The latter is more stable when a large number of roots are requested.

Properties in the RHF EOM implementation

The only property that can be calculated with the current RHF EOM implementation is the transition moment. It is calculated as a CI-like expectation value, as proposed by Stanton and Bartlett. The right and left transition density are defined as

$$\rho_{pq}^{Gr\to Ex} = \langle\phi_0|(1+\Lambda)[e^{-T}\{p^+q^-\}e^T, R]|\phi_0\rangle$$

$$\rho_{pq}^{Ex\to Gr} = \langle\phi_0|Le^{-T}\{p^+q^-\}e^T|\phi_0\rangle$$

In the above equation, Λ corresponds to the ground state left vector, which needs to be solved once and L is the left vector, which needs to be solved separately for each root. Once the right and left vectors have been obtained, the left and right transition densities are constructed and the oscillator strength is calculated using the following formula

$$f = \frac{2}{3}\varepsilon|\mu_{pq}\rho_{pq}^{Ex\to Gr}||\mu_{pq}\rho_{pq}^{Gr\to Ex}|$$

The oscillator strength, calculated by default, employs a linear approximation for Λ. The L vectors are, on the other hand, calculated as a general inverse of the corresponding R vectors. This approximation requires no additional effort

over the energy calculation and gives similar accuracy as that of the exact oscillator strength calculation, which is at least twice the cost of the energy calculation. Exact EOM-CC transition moments can, however, be calculated by setting `DoLeft` and `DoTDM` to true. Please note that transition moments have not yet been implemented for the UHF EOM-CCSD approach.

5.9.2 General Use

The EOM-CCSD routine is part of the `orca_mdci` module of the ORCA program package. It is called after a successful coupled-cluster calculation, if the appropriate flags and the number of roots have been set. In the following chapter, the general program flow and the input parameters of the EOM routine will be described. For an RHF or UHF reference, the EE-, IP- and EA-EOM-CCSD approaches are available for the computation of excitation energies, ionization potentials and electron affinities, respectively. Currently, the following simple input keywords are available:

```
!EOM-CCSD      # same as !EE-EOM-CCSD
!EE-EOM-CCSD   # EOM for electronically excited states
!IP-EOM-CCSD   # IP version
!EA-EOM-CCSD   # EA version
```

The simplest way to perform an EOM calculation is via the usage of the `EOM-CCSD` keyword, together with the specification of the desired number of roots:

```
! RHF EOM-CCSD cc-pVDZ TightSCF

%mdci
  nroots 9
end

*xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

The above input will call the EOM routine with default settings. The main output is a list of excitation energies, augmented with some further state-specific data. For the above input, the following output is obtained:

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1:  0.147823 au     4.022 eV   32443.5 cm**-1
  Amplitude    Excitation
   0.107945     4 ->   8
   0.665496     7 ->   8
   0.104633     7 ->   8     6 ->   8
  Ground state amplitude:  0.000000
Percentage singles character=     92.32

IROOT=  2:  0.314133 au     8.548 eV   68944.3 cm**-1
  Amplitude    Excitation
   0.671246     7 ->   9
  Ground state amplitude: -0.000000
Percentage singles character=     90.42

IROOT=  3:  0.343833 au     9.356 eV   75462.6 cm**-1
  Amplitude    Excitation
  -0.670633     5 ->   8
  -0.112538     6 ->   8     5 ->   8
```

(continues on next page)

(continued from previous page)

```
  Ground state amplitude:  0.000000
Percentage singles character=     92.00

IROOT=  4:  0.364199 au     9.910 eV   79932.5 cm**-1
  Amplitude    Excitation
   0.102777     4 ->  10
  -0.484661     6 ->   8
   0.438311     7 ->  10
  -0.167512     6 ->   8     6 ->   8
  Ground state amplitude: -0.021060
Percentage singles character=     87.22

IROOT=  5:  0.389398 au    10.596 eV   85463.0 cm**-1
  Amplitude    Excitation
   0.646812     4 ->   8
  -0.122387     7 ->   8
   0.171366     7 ->   8     6 ->   8
  Ground state amplitude:  0.000000
Percentage singles character=     87.47

IROOT=  6:  0.414587 au    11.281 eV   90991.4 cm**-1
  Amplitude    Excitation
  -0.378418     6 ->   8
  -0.537292     7 ->  10
  -0.124246     6 ->   8     6 ->   8
  Ground state amplitude: -0.061047
Percentage singles character=     89.13

IROOT=  7:  0.423861 au    11.534 eV   93026.7 cm**-1
  Amplitude    Excitation
   0.673806     7 ->  11
  Ground state amplitude:  0.000000
Percentage singles character=     93.14

IROOT=  8:  0.444201 au    12.087 eV   97490.8 cm**-1
  Amplitude    Excitation
   0.664877     6 ->   9
   0.130475     6 ->   9     6 ->   8
  Ground state amplitude: -0.000000
Percentage singles character=     87.17

IROOT=  9:  0.510514 au    13.892 eV  112044.8 cm**-1
  Amplitude    Excitation
  -0.665791     6 ->  10
   0.114259     6 ->  15
  -0.124374     6 ->  10     6 ->   8
  Ground state amplitude: -0.000000
```

The IP and EA versions can be called using the keywords `IP-EOM-CCSD` and `EA-EOM-CCSD`, respectively. For open-shell systems (UHF reference wavefunction), IP/EA-EOM-CCSD calculations require an additional keyword. Namely, an IP/EA calculation involving the removal/attachment of an α electron is requested by setting the `DoAlpha` keyword to true in the %mdci block, while setting the `DoBeta` keyword to true selects an IP/EA calculation for the removal/attachment of a β electron. Note that `DoAlpha` and `DoBeta` cannot simultaneously be true and that the calculation defaults to one in which `DoAlpha` is true if no keyword is specified on input. A simple example of the input for a UHF IP-EOM-CCSD calculation for the removal of an α electron is given below.

```
! IP-EOM-CCSD cc-pVDZ
%mdci
  DoAlpha true
  NRoots 7
end
```

(continues on next page)

(continued from previous page)

```
*xyz 0 3
  O      0.0 0.0 0.0
  O      0.0 0.0 1.207
*
```

Capabilities

At present, the EOM routine is able to perform excited, ionized and electron attached state calculations, for both closed- or open-shell systems, using RHF or UHF reference wave functions, respectively. It can be used for serial and parallel calculations. The method is available in the back-transformed PNO and DLPNO framework enabling the calculation of large molecules - see Section *Excited States via PNO-based coupled cluster* and Section *Excited States with DLPNO based coupled cluster methods*. In the closed-shell case (RHF), a lower-scaling version can be invoked by setting the `CCSD2` keyword to true in the %mdci section. The latter is a second-order approximation to the conventional EOM-CCSD. For the time being, the most useful information provided is the list of the excitation energies, the ionization potentials or the electron affinities. The ground to excited state transition moments are also available for the closed-shell implementation of EE-EOM-CCSD.

5.9.3 Memory Management

The most important data coming from the coupled-cluster routine are the ground state energy and wave function, and the molecular integrals. The integrals are then used to create "dressed" integral containers, which allows for an efficient factorization of the EOM equations, since these dressed quantities do not change during the calculation. Most of these are written on disk, with the possible exception of the integral container which has three external labels. This, and the solver files may remain in core if enough memory is available. The program sequentially tries to allocate memory for the files in the order of their importance, and what cannot be kept in core, goes on disk. The order of allocation is as follows: 1. residual vectors, 2. Ritz vectors, 3. three external integrals, 4. sigma vectors and 5. state (trial) vectors, as seen in the example below:

```
--------------------------------
AUTOMATIC CHOICE OF INCORE LEVEL
--------------------------------

Memory available                          ...   6512.00 MB
Memory needed for Residual-vectors        ...     71.27 MB
Memory needed for Ritz-vectors            ...     71.27 MB
Memory needed for 3-ext integrals         ...     92.05 MB
Memory needed for Sigma-vectors           ...   1425.31 MB
Memory needed for State-vectors           ...   1425.31 MB
 -> Final InCoreLevel    ... 5
```

Half of the memory specified with the keyword `MaxCore` is distributed among the five candidates. In the above case, everything fits in memory. Note that these are only the largest contributors to memory consumption, and there should ideally be a safety margin when allocating memory.

In order to estimate the amount of necessary memory, it should be kept in mind that, in the closed shell case, the memory requirements of the residual and Ritz vectors are proportional to $N_R N_P N_V^2$, the three external integrals to $N_R N_O N_V^3$ and the sigma and trial vectors to $N_D N_R N_P N_V^2$, where N_O and N_V are the number of occupied and virtual orbitals, $N_P = \frac{N_O(N_O+1)}{2}$ is the number of occupied pairs, N_R is the number of roots, and N_D is the maximum size of the reduced space. The keyword `NRoots` sets N_R, while `NDav` determines N_D. Luckily, the contributions that, in our experience, are the most important to keep in memory, are also the ones that require the smallest amount of it. It is advisable to use `KCOpt AOBLAS`, as it has the lowest memory requirements.

Note that in the UHF EE-EOM-CCSD implementation, the memory requirements of the residual and Ritz vectors are proportional to $N_R(N_{P_\alpha} N_{V_\alpha}^2 + N_{P_\beta} N_{V_\beta}^2 + N_{O_\alpha} N_{O_\beta} N_{V_\alpha} N_{V_\beta})$, the three external integrals to $N_R(N_{O_\alpha} N_{V_\alpha}^2 + N_{O_\beta} N_{V_\beta}^2 + N_{O_\alpha} N_{V_\alpha} N_{V_\beta}^2 + N_{O_\beta} N_{V_\beta} N_{V_\alpha}^2)$ and the sigma and trial vectors memory requirements are proportional to $N_D N_R(N_{P_\alpha} N_{V_\alpha}^2 + N_{P_\beta} N_{V_\beta}^2 + N_{O_\alpha} N_{O_\beta} N_{V_\alpha} N_{V_\beta})$, where N_{O_α}, N_{O_β}, N_{V_α} and N_{V_β} are respectively, the

number of occupied alpha, occupied beta, virtual alpha and virtual beta orbitals and $N_{P_\alpha} = \frac{N_{O_\alpha}(N_{O_\alpha}-1)}{2}$ and $N_{P_\beta} = \frac{N_{O_\beta}(N_{O_\beta}-1)}{2}$ are the number of alpha and beta occupied pairs, respectively.

5.9.4 Some tips and tricks for EOM-CC calculation

- The COSX approximation gives significant savings in terms of memory use, disk space use and computational timings with almost no loss of accuracy[662]. Therefore, the preferred setting for large-scale calculations should include `DoCOSXEOM true`, `DoAOX3e true` and `KCOpt KC_AOBLAS` (Note that `KC_AOX` is the only option available for `KCOpt` in the UHF implementation).
- The EOM-CC code in ORCA has three versions of the Davidson solver. The default one is the multi-root solver which does optimization of all the roots together. It gives the fastest convergence and is more suitable when one is interested only in a few roots of a big molecule. However, the multi-root solver can run into numerical issues, if more than 10 roots are desired. In that case, one can invoke the root-wise solver by setting `DoRootwise true`. The single root solver is very stable and should be used when a large number of roots are desired. However, the convergence of the single root solver is slower than the multi-root one. In the RHF implementation, there is also a batchwise solver, where a subset of the total number of roots is optimized together. This can be invoked by setting `NRootsPerBatch true` and is intermediate between the multi-root and single-root solvers in terms of stability and convergence.
- If the EOM iterations do not converge within 50 cycles, one can try to increase the number of iterations by setting `MaxIter` in the `%mdci` block to a larger value. One can also try to increase the dimension of the Krylov space in the Davidson solver by increasing the `NDav` value and this generally helps to accelerate convergence. However, setting `NDav` to a value larger than 200 can make the calculation prohibitively costly especially in terms of memory.
- Convergence thresholds of `DTol 1e-5` (Davidson convergence) and `STol 1e-7` (ground state CCSD convergence) generally yield sufficiently converged energies, and are suitable for most purposes.
- The normal Davidson solver generally leads to the lowest-energy solutions. This procedure can also yield roots dominated by double excitations (the so-called satellite states) for the IP and EA variants of EOM-CC, when one asks for a large number of roots. If one is interested in the low-lying Koopman's type of IP and EA states, they can be obtained by setting `FollowCIS` to true. This will follow the initial guess provided by the Fock operators.

5.9.5 Full keyword list

The simple input keywords related to EOM-CC are collected in Table 5.10. The input keywords of the `%mdci` block related to UHF EOM-CC are collected in Table 5.12. The input keywords of the `%mdci` block related to RHF EOM-CC are collected in Table 5.11.

Table 5.10: Simple input keywords for EOM-CC.

Keyword	Description
`EOM-CCSD`	Same as `EE-EOM-CCSD`
`EE-EOM-CCSD`	EOM-CC for electronically excited states
`IP-EOM-CCSD`	IP version
`EA-EOM-CCSD`	EA version

Table 5.11: `%mdci` block input keywords for RHF EOM-CC with defaults in parentheses.

Keyword	Options	Description
`DoEOM` (false)	`true/false`	Perform EOM?
`UseEOMOptS` (true)	`true/false`	Use optimized sigma routines for singles?
`UseEOMOptD` (true)	`true/false`	Use optimized sigma routines for doubles?
`NDav` (20)	`<int>`	Max. size of reduced space (i.e. 20*NRoots)

continues on next pag

Table 5.11 – continued from previous page

Keyword	Options	Description
CCSD2 (false)	true/false	Use the lower scaling CCSD(2) approximation?
CheckEachRoot (true)	true/false	Check convergence for each root separately?
RootHoming (true)	true/false	Apply root homing?
DoLanczos (false)	true/false	Use the Lanczos procedure rather than Davidson?
UseCISUpdate (true)	true/false	Use diagonal CIS for updating?
NInits (0)	<int>	Number of roots in the initial guess, if 0, use preset value
DRESS3ES (true)	true/false	Construct the external dressing to singles or calculate on the fly?
DRESS3ED (false)	true/false	Construct the external dressing to doubles or calculate on the fly?
DOCOSXEOM (false)	true/false	Use COSX approximation for external exchange term in EOM?
DOAOX3E (false)	true/false	Use COSX approximation for 4 external terms contribution to 3 external intermediat
DoRootwise (false)	true/false	Solve for each root separately? More stable for large number of roots
DoTDM (false)	true/false	Option for calculation of default transition moment?
Doleft (false)	true/false	Calculation of exact left vector?
NRootsPerBatch (1)	<int>	Number of roots calculated together
FOLLOWCIS (false)	true/false	Follows the initial singles guess?
DoCore (true)	true/false	Initiates ionization or excitation from core orbital?
CoreHole (0)	<int>	Core orbital from which ionization or excitation is needed
CVSEP (false)	true/false	Separate core orbitals from valence?
DTol (1e-5)	<real>	Default for EOM residual threshold
NRoots (9)	<int>	Number of roots
OTol (1e-14)	<real>	Orthogonalization threshold
KCOpt		Method for external exchange formation
	KC_MO	
	KC_AOX	When asked for exact TDM calculation
	KC_AOBLAS	Most efficient
PrintLevel (3)	<int>	The amount of information to be printed
MaxCore (500)	<int>	Total amount of memory

Table 5.12: %mdci block input keywords for UHF EOM-CC with defaults in parentheses.

Keyword	Options	Description
DoEOM (false)	true/ false	Perform EOM?
DoAlpha (false)	true/ false	Removal/attachment of an alpha electron (IP/EA calculations)?
DoBeta (false)	true/ false	Removal/attachment of a beta electron (IP/EA calculations)?
NDav (20)	<int>	Max. size of reduced space (i.e. 20*NRoots)
UseQROs (false)	true/ false	Use QROs?
CheckEachRoot (true)	true/ false	Check convergence for each root separately?
RootHoming (true)	true/ false	Apply root homing?
DoLanczos (false)	true/ false	Use the Lanczos procedure rather than Davidson?
DoOlsen (false)	true/ false	Use the Olsen procedure rather than Davidson?
UseCISUpdate (true)	true/ false	Use diagonal CIS for updating?
NInits (0)	<int>	Number of roots in the initial guess, if 0, use preset value
DOCOSXEOM (false)	true/ false	Use COSX approximation for external exchange term in EOM?
DOAOX3E (false)	true/ false	Use COSX approximation for 4 external terms contribution to 3 external intermediate?
DoRootwise (false)	true/ false	Solve for each root separately? More stable for large number of roots
NRootsPerBatch (1)	<int>	Number of roots calculated together
FOLLOWCIS (true)	true/ false	Follow the initial singles guess?
DTol (1e-5)	<real>	Default for EOM residual threshold
NRoots (9)	<int>	Number of roots
OTol (1e-14)	<real>	Orthogonalization threshold
KCOpt	KC_AOX	AO exchange for the four external contributions
PrintLevel (3)	<int>	The amount of information to be printed
MaxCore (500)	<int>	Total amount of memory

5.10 Excited States via STEOM-CCSD

The EOM-CCSD approach for excitation energies becomes prohibitively costly for large systems because of its O(N^6) scaling. Therefore, one needs a more compact form of the wave-function ansatz. A second similarity transformation can compress the final matrix diagonalization step to the CIS space only. The resulting STEOM-CCSD method of Marcel Nooijen and co-workers [663] is an efficient way for accurate calculations of excitation energies.

5.10.1 General Description

In the standard EOM-CC method, the transformed Hamiltonian is diagonalized over a singles and doubles space to obtain ionized, attached, or excited states of the reference state. In STEOM-CC, one performs a second similarity transformation

$$\hat{G} = \{e^{\hat{S}}\}^{-1}\hat{\bar{H}}\{e^{\hat{S}}\}$$

The transformation operator $\hat{S}$, including singles and doubles, is defined as

$$\hat{S} = \hat{S}^{IP} + \hat{S}^{EA},$$

$$\hat{S}^{IP} = S_{i'}^{m}\hat{E}_{i'}^{m} + \frac{1}{2}S_{ij}^{mb}\hat{E}_{ij}^{mb},$$

$$\hat{S}^{EA} = S_{e}^{a'}\hat{E}_{e}^{a'} + \frac{1}{2}S_{ej}^{ab}\hat{E}_{ej}^{ab}.$$

In the above equations, m and e denote active indices of the hole and particle type respectively, while a prime denotes a restriction to orbitals that are not active. The amplitudes of the operator $\hat{S}$ are defined in such a way that matrix elements of the transformed Hamiltonian, in second quantized notation, become equal to zero.

$$g_{i'}^{m} = g_{ij}^{mb} = g_{e}^{a'} = g_{ej}^{ab} = 0$$

In addition, the zeros which pre-existed in $\bar{H}$, after solving the CCSD equations, remain preserved. The above equations are linear in $\hat{S}$ and are equivalent to the Fock space multireference coupled cluster equations for the one valence problem. However, to ensure numerical stability, the equations are re-casted as matrix diagonalization problem and solved as IP-EOM-CCSD and EA-EOM-CCSD problems. The $\hat{S}^{IP}$ and $\hat{S}^{EA}$ are then extracted from the converged previous calculations, respectively, by invoking intermediate normalization on the suitably chosen eigenvectors corresponding to active holes and active particles. The total process can be described as following

- Solution of the ground state coupled cluster equations
- Construct the first similarity transformed Hamiltonian as $\hat{\bar{H}} = e^{-\hat{T}}\hat{H}e^{\hat{T}}$
- Solution of the IP-EOM and EA-EOM equations
- Extraction of the $\hat{S}$ amplitudes
- Construct the second similarity transformed Hamiltonian as $\hat{\bar{G}} = e^{-\hat{S}}\hat{\bar{H}}e^{\hat{S}}$
- Diagonalization of $\hat{\bar{G}}$ in CIS space

The advantage of the above method is that, instead of one iterative O(N^6) scaling diagonalization step, it requires two iterative O(N^5) scaling steps, one non-iterative O(N^5) scaling step and one iterative O(N^4) scaling matrix diagonalization step. The presence of so-called 'implicit triples excitation' term ensures the charge transfer separability of the excited states, which is absent in EOM-CCSD. In addition, since the final diagonalization step is performed in a CIS space, the spin adaption is trivial and excited states of triplet multiplicity can be obtained without going through the complications of a spin orbital based implementation.

The STEOMCC approach has also recently been extended for applications to open-shell systems within the UHF formalism [664] [665]. In this case, the expressions for the operators $\hat{S}^{IP}$ and $\hat{S}^{EA}$ take the form,

$$\begin{aligned}
\hat{S}^{IP} &= \frac{1}{2}\sum_{i,e,a,b} s_{ie}^{ab}\{\hat{a}^\dagger\,\hat{b}^\dagger\,\hat{e}\,\hat{i}\} + \sum_{\bar{i},e,\bar{a},b} s_{\bar{i}e}^{\bar{a}b}\{\hat{\bar{a}}^\dagger\,\hat{b}^\dagger\,\hat{e}\,\hat{\bar{i}}\} \\
&+ \frac{1}{2}\sum_{\bar{i},\bar{e},\bar{a},\bar{b}} s_{\bar{i}\bar{e}}^{\bar{a}\bar{b}}\{\hat{\bar{a}}^\dagger\,\hat{\bar{b}}^\dagger\,\hat{\bar{e}}\,\hat{\bar{i}}\} + \sum_{i,\bar{e},a,\bar{b}} s_{i\bar{e}}^{a\bar{b}}\{\hat{a}^\dagger\,\hat{\bar{b}}^\dagger\,\hat{\bar{e}}\,\hat{i}\}, \\
\hat{S}_{-} &= \frac{1}{2}\sum_{i,j,a,m} s_{ij}^{am}\{\hat{a}^\dagger\,\hat{m}^\dagger\,\hat{j}\,\hat{i}\} + \sum_{\bar{i},j,\bar{a},m} s_{\bar{i}j}^{\bar{a}m}\{\hat{\bar{a}}^\dagger\,\hat{m}^\dagger\,\hat{j}\,\hat{\bar{i}}\} \\
&+ \frac{1}{2}\sum_{\bar{i},\bar{j},\bar{a},\bar{m}} s_{\bar{i}\bar{j}}^{\bar{a}\bar{m}}\{\hat{\bar{a}}^\dagger\,\hat{\bar{m}}^\dagger\,\hat{\bar{j}}\,\hat{\bar{i}}\} + \sum_{i,\bar{j},a,\bar{m}} s_{i\bar{j}}^{a\bar{m}}\{\hat{a}^\dagger\,\hat{\bar{m}}^\dagger\,\hat{\bar{j}}\,\hat{i}\}.
\end{aligned}$$

where we use overbars to distinguish the β orbitals from the α orbitals. The amplitudes $\left\{s_{i\bar{e}}^{ab}, s_{\bar{i}e}^{\bar{a}b}\right\}$ are determined by solving the UHF EA-EOM-CCSD equations for the attachment of an α electron, while the $\left\{s_{\bar{i}\bar{e}}^{\bar{a}\bar{b}}, s_{i\bar{e}}^{a\bar{b}}\right\}$ amplitudes are extracted from a UHF EA-EOM-CCSD calculation for the attachment of a β electron. Similarly, the sets of amplitudes $\left\{s_{ij}^{am}, s_{\bar{i}j}^{\bar{a}m}\right\}$ and $\left\{s_{\bar{i}\bar{j}}^{\bar{a}\bar{b}}, s_{i\bar{j}}^{a\bar{m}}\right\}$ are determined by solving the decoupled UHF IP-EOM-CCSD problems for the ionization of an α electron and the ionization of a β electron, respectively. Hence, an UHF STEOMCC calculation involves two separate IP calculations (O(N^5) scaling) and two separate EA calculations (O(N^5) scaling steps).

All the speed up options, including CCSD(2) (only available in RHF implementation) and COSX, which are available for EOM-CCSD are also available for STEOMCC. The most important steps in a STEOMCC calculation are the IP-EOM and EA-EOM calculations. These steps are performed using the EOM-CCSD module and the relevant keywords are the same as that described in *Excited States via EOM-CCSD*. The keywords which are exclusive to the RHF STEOM module are:

```
%mdci
#RHF STEOM parameters - defaults displayed
DoCISNat true         # automatic selection of active space
NActIP  3             # number of states defined as active in the IP calculation
NActEA  2             # number of states defined as active in the EA calculation
DoTriplet false       # target state of triplet multiplicity
DoDbFilter true       # filters out states with doubles excitation  character
DoNewActSch true     # new active space selection scheme for STEOM-CCSD
DoSOLV                # perturbative correction for solvation effects (experimental)
#Default values for automatic active space selection scheme
OThresh 0.001         # Cut off occupation of CIS natural orbitals in IP calculation
VThresh 0.001         # Cut off occupation of CIS natural orbitals in EA calculation
IPSThrs 80             #  The percentage singles threshold for the IP calculation
EASThrs 80            #  The percentage singles threshold for the EA calculation
end
```

The keywords pertaining to the UHF STEOM module are:

```
%mdci
#UHF STEOM parameters - defaults displayed
DoCISNat true         # automatic selection of active space
NActIP_a  3           # number of states defined as active in the IP calculation
                      # for the removal of an α electron
NActIP_b  3           # number of states defined as active in the IP calculation
                      # for the removal of a β electron
NActEA_a  2           # number of states defined as active in the EA calculation
                      # for the attachment of an α electron
NActEA_b 2            # number of states defined as active in the EA calculation
                      # for the attachment of a β electron
DoDbFilter true       # filters out states with doubles excitation  character
UseQROs  false        # use QROs or not
DoNewActSch true      # new active space selection scheme for STEOM-CCSD
#Default values for automatic active space selection scheme
OThresh 0.001         # Cut off occupation of CIS natural orbitals in tIP␣
↪calculations
VThresh 0.001         # Cut off occupation of CIS natural orbitals in EA␣
↪calculations
IPSThrs 80            #  The percentage singles threshold for the IP calculations
EASThrs 80            #  The percentage singles threshold for the EA calculations
end
```

5.10.2 Selection of Active space

The results of a STEOM-CC calculation depend upon the number of roots selected as active in the EOMIP and EOMEA calculations. In ORCA, they are chosen automatically, by using state-averaged CIS natural transition orbitals (NTO). By default, the number of roots included in this initial CIS computation is equal to the number of roots requested in STEOM (`NRoots`). However, this can be modified setting `NRootsCISNAT` to higher values. The orbitals up to a predefined occupation are then chosen to be active in the EOMIP and EOMEA calculations, and this is controlled by the keywords `OThresh` and `VThresh` respectively. Now, there are two possible ways to chose active space. One is to use the criteria of percentage occupation of NTO's as described in ref [666]. However, a newer and more robust approach is to use the criteria of absolute occupation, which is default in the current implementation. One can switch on the old percentage occupation based active space selection by setting `DoNewActSch` to false (not recommended).

One can also select the active spaces manually by turning the `DoCISNat` to false and setting the `NActIP` and `NActEA` (RHF STEOM calculation) or the `NActIP_a`, `NActIP_b`, `NActEA_a` and `NActEA_b` (UHF STEOM calculation) to desired values. However, this is not recommended for general uses. The following snippet shows the output of the active orbital selection procedure on a closed-shell molecule:

```
-----------------------------------------
STATE AVERAGED NATURAL ORBITALS FOR ACTIVE SPACE SELECTION
-----------------------------------------

Solving eigenvalue problem for the occupied space  ... Occupied block occupation :
     0   0.000478
     1   0.002266
     2   0.169928
     3   0.171663
     4   0.310125
     5   0.345541
Orbital taken as active for IP roots:
     0   0.345541
     1   0.310125
     2   0.171663
     3   0.169928
done
Solving eigenvalue problem for the virtual space   ... Virtual block occupation :
     6   0.640886
     7   0.332262
     8   0.017272
     9   0.005326
    10   0.001752
    11   0.000667
    12   0.000574
    13   0.000540
    14   0.000160
    15   0.000150
    16   0.000139
    17   0.000086
    18   0.000082
    19   0.000037
    20   0.000023
    21   0.000016
    22   0.000013
    23   0.000008
    24   0.000003
    25   0.000002
    26   0.000001
    27   0.000000
    28   0.000000
    29   0.000000
    30   0.000000
    31   0.000000
```

(continues on next page)

(continued from previous page)

```
    32   0.000000
    33   0.000000
    34   0.000000
    35  -0.000000
Orbital taken as active for EA roots :
     0   0.640886
     1   0.332262
     2   0.017272
done
 No of roots active in IP calculation:    4
 No of roots active in EA calculation:    3
```

5.10.3 Active space selection using TD-DFT densities

Instead of using a CIS calculation for selected the Active Space roots, a TD-DFT based one can also be considered. Be aware that using DFT Kohn-Sham orbitals for computing the CCSD GS energy can lead to some instabilities and give incorrect results.

The main interest of this approach is to start the STEOM-CCSD calculation with TD-DFT electronic densities which are in general better than the CIS one, especially for some specific compounds (metallic complexes for example). The computed TD-DFT densities are also often more stable than the CIS one. It will however slow down the calculation.

The input has to be written like this:

```
!RHF/UHF BHANDHLYP STEOM-CCSD TZVP

%mdci
nroots 10
tddftguess true
end

%tddft
nroots 10
end

*xyz 0 1
```

Any DFT functional can be used but we recommend one with a decent amount of HF exchange. On top of this, the keyword `TDDFTGuess` has to be set to true in mdci block and the tddft has to be added together with the `NRoots` keyword. In both input blocks (`%mdci` and `%tddft`) the same number of roots has to be given. Starting from ORCA 6, using TD-DFT guess with an UHF reference is also possible.

5.10.4 The reliability of the calculated excitation energy

The excitation energy for any states calculated in STEOM-CC are only reliable when the dominant excitation for that states are confined within the active space. This can be verified from the percentage active character of the calculated states, an *a posteriori* diagnostic which is defined as

$$\%activecharacter = \frac{\sum\limits_{m,e} C(m,e) * C(m,e)}{\sum\limits_{i,a} C(i,a) * C(i,a)} * 100$$

for closed-shell systems and takes the form,

$$\%activecharacter = \frac{\sum\limits_{m,e} C(m,e) * C(m,e) + \sum\limits_{\bar{m},\bar{e}} C(\bar{m},\bar{e}) * C(\bar{m},\bar{e})}{\sum\limits_{i,a} C(i,a) * C(i,a) + \sum\limits_{\bar{i},\bar{a}} C(\bar{i},\bar{a}) * C(\bar{i},\bar{a})} * 100.$$

within the UHF formalism. The roots which have $\%activecharacter$ higher than 98.0 are considered to be converged with respect to the active space.

```
------------------
STEOM-CCSD RESULTS
------------------

IROOT=  1:  0.145412 au     3.957 eV   31914.3 cm**-1
  Amplitude    Excitation
  -0.169361     4 ->   8
  -0.984822     7 ->   8

Percentage Active Character     99.86

  Amplitude    Excitation in Canonical Basis
  -0.166580     4 ->   8
  -0.975432     7 ->   8
  -0.124356     7 ->  13

IROOT=  2:  0.309409 au     8.419 eV   67907.5 cm**-1
  Amplitude    Excitation
   0.994141     7 ->   9

Percentage Active Character     99.78

  Amplitude    Excitation in Canonical Basis
  -0.990029     7 ->   9

IROOT=  3:  0.336993 au     9.170 eV   73961.4 cm**-1
  Amplitude    Excitation
  -0.994078     5 ->   8

Percentage Active Character     99.10

  Amplitude    Excitation in Canonical Basis
  -0.984116     5 ->   8
  -0.136769     5 ->  13

IROOT=  4:  0.357473 au     9.727 eV   78456.2 cm**-1
  Amplitude    Excitation
   0.181761     4 ->  10
   0.728209     6 ->   8
   0.611668     7 ->  10
  -0.191540     7 ->  12

Percentage Active Character     94.10

Warning:: the state may have not converged with respect to active space
-------------------- Handle with Care --------------------

  Amplitude    Excitation in Canonical Basis
  -0.184144     4 ->  10
  -0.725183     6 ->   8
  -0.633718     7 ->  10

IROOT=  5:  0.386654 au    10.521 eV   84860.8 cm**-1
  Amplitude    Excitation
   0.980406     4 ->   8
  -0.178551     7 ->   8

Percentage Active Character     99.79
```

(continues on next page)

(continued from previous page)

```
  Amplitude    Excitation in Canonical Basis
   0.971678     4 ->   8
   0.122877     4 ->  13
  -0.179242     7 ->   8

IROOT=  6:  0.444881 au    12.106 eV   97640.1 cm**-1
  Amplitude    Excitation
  -0.995150     6 ->   9

Percentage Active Character     99.69

  Amplitude    Excitation in Canonical Basis
  -0.989966     6 ->   9
```

If the *%activecharacter* for any calculated state is less than 98, that state may have not converged with respect to active space and the excitation energy for that particular state is less reliable. The user should request a larger number of roots under those conditions.

5.10.5 Removal of IP and EA states with double excitation character

To obtain accurate results with STEOM-CCSD, only the $\hat{S}$ amplitudes corresponding to the states dominated by single excitations should be included in the second similarity transformation. This is ensured in ORCA in two ways. First, the root following (`FollowCIS`) is activated by default so that it converges to the states dominated by singly excited guess vectors. This avoids the calculation of so called 'satellite states', which are of double excitation character with respect to the ground state. Secondly, among the converged IP and EA roots, the states which have %singles character below a certain predefined threshold (i.e. controlled by the keywords `IPThresh` and `EAThresh`) are automatically excluded from the second similarity transformation.

```
EOM-CCSD RESULTS
----------------

IROOT=  1:  0.105316 au     2.866 eV   23114.2 cm**-1
  Amplitude    Excitation
   0.697547     x ->   8
IROOT=  2:  0.217925 au     5.930 eV   47829.1 cm**-1
  Amplitude    Excitation
  -0.701454     x ->   9
IROOT=  3:  0.304098 au     8.275 eV   66741.8 cm**-1
  Amplitude    Excitation
  -0.700458     x ->  10
IROOT=  4:  0.350387 au     9.535 eV   76901.1 cm**-1
  Amplitude    Excitation
   0.702705     x ->  11
IROOT=  5:  0.651462 au    17.727 eV  142979.4 cm**-1
  Amplitude    Excitation
   0.637352     x ->  12
   0.121747     x ->   8     4 ->  10
   0.177039     x ->   8     5 ->   9
   0.109987     x ->   9     5 ->   8
  -0.206789     x ->   8     7 ->  10
  -0.109870     x ->  10     7 ->   8

EA STATE=  0:  percentage singles     95.282
EA STATE=  1:  percentage singles     96.981
EA STATE=  2:  percentage singles     96.540
EA STATE=  3:  percentage singles     97.844
EA STATE=  4:  percentage singles     68.884
```

(continues on next page)

(continued from previous page)

```
Warning: high double excitation character, excluding from the STEOM transformation
 Final no active EA roots:    4
```

Note that the use of CIS natural transition orbitals can lead to convergence issues for the IP and EA states which are dominated by double excitation character. This can be remedied by setting `DoDbFilter` to true.

5.10.6 Transition and difference densities

At the end of a STEOM computation, it is possible to store the final eigenvectors in a file "*job.cis*", in analogy with what is done for CIS and TD-DFT computations. This file can be obtained by setting `DoStoreSTEOM true` in the input. This file can then be processed by `orca_plot` to obtain the difference and / or the transition densities.

An Natural Transition Orbitals analysis can also be performed within the STEOM-CCSD scheme, as described in *Natural Transition Orbitals*. It can be performed by setting the keyword `DoSTEOMNatTransOrb` to true.

5.10.7 Properties

The dipolar and transition moments (as well as the oscillator strength) can be computed within the STEOM module using different kinds of approximations. Please cite our paper on these corrected STEOM transition densities [667]! Starting from ORCA 5, new defaults (`DoSimpleDens false`) are used that are much better than the previous CIS-like approximation, and the full option is of CC3-like quality.

```
%mdci

DosimpleDens false      # Default, using the STEOM-CCSD density + some doubles␣
↪effect.
AddL2term true

DosimpleDens false      # using the STEOM-CCSD density + some doubles effect.
AddL2term true          # + neglected GS double
UpdateL1 true

DosimpleDens false      # using the STEOM-CCSD density + some doubles effect.
AddL2term true          # + neglected GS double + doubles from EOM-CCSD
UpdateL1 true           # (expensive, but of CC3 quality - see reference)
AddDDTerm true

end
```

By default, the STEOM-CCSD densities with `AddL2term true` should be used for all calculation as discussed in ref. [667].

5.10.8 Solvation (Experimental)

In STEOM-CCSD, the excitation energies and densities can be corrected using the CPCM solvation scheme in ORCA.

To use it, the keyword `DoSolv` has to be set to true in the `%mdci` block and the simple keyword `CPCM` (or `SMD`) + name of the solvent has to be given.

```
!CPCM(ethanol) STEOM-CCSD TightSCF def2-TZVP def2-TZVP/C def2/J

%mdci
Nroots 5
DoSolv true
end
```

5.10.9 Spin-Orbit Coupling (Experimental)

You can compute the spin-orbit coupling between singlet and triplets states in STEOM-CCSD using the keyword `STEOMSOC true`. Please note that all SOC matrix elements and properties are currently computed from the **right vector only**!

5.10.10 Core excitation

The STEOM-CCSD (and bt-PNO-STEOM-CCSD) method can also be used to compute the K-edge core-excitation energy of molecules. See *Core-Excitation* for more details.

5.10.11 Transient absorption

Transient absorption spectra can be computed using the keyword `DoTrans true`. The `IRoot` keyword will select the targeted excited state.

5.11 Excited States via IH-FSMR-CCSD

An alternative approach for decoupling the singles excitation space from the space of double and higher excitations is to use the so called Fock space multi-reference coupled cluster (FSMRCC) method. The method is similar to STEOM-CCSD, but much more flexible in terms of formulation.

5.11.1 General Description

FSMRCC is originally based on an effective Hamiltonian (EH). The basic idea of EH theory is to obtain some selective eigenvalues of the Hamiltonian operator from the total eigenvalue spectrum. For this purpose, the entire configuration space is divided into a model and an outer space with projection operators P_M and Q_M, respectively (see Fig. 5.30). The diagonalization of the EH takes care of the non-dynamic correlation coming from the interactions between the model space configurations. On the other hand, the dynamic correlation arises due to the interactions of the model space configurations with the outer space configurations. This interaction is introduced through a universal wave operator Ω, which is parametrized such that it generates the exact wave function when acting on the model space. The valence universal wave operator Ω has the form

$$\Omega = e^{\tilde{S}^{(p,h)}}$$

where the braces indicate normal ordering of the cluster operators and $\tilde{S}^{(p,h)}$ is defined as

$$\tilde{S}^{(p,h)} = \sum_{k=0}^{p} \sum_{l=0}^{h} \tilde{S}^{(k,l)}$$

The cluster operator $\tilde{S}^{(k,l)}$ is capable of destroying exactly k active particles and l active holes, in addition to creation of holes and particles. The $\tilde{S}^{(p,h)}$ subsumes all lower sector Fock space $\tilde{S}^{(k,l)}$ operators. The $\tilde{S}^{(0,0)}$ is equivalent to standard single-reference coupled cluster $\hat{T}$ operator. The EH for (p,h) valence system can be defined as

$$\hat{H}_{eff} = P_M^{(p,h)} \Omega^{-1} \hat{H} \Omega P_M^{(p,h)}$$

However, Ω^{-1} may not be well defined in all the cases. Therefore, the above definition for the EH is seldom used. Instead, the Block-Lindgren approach is generally used for solving the equations, which is defined by

$$P_M^{(p,h)} \left[\hat{H}\Omega - \Omega \hat{H}_{eff} \right] P_M^{(p,h)} = 0$$

$$Q_M^{(p,h)} \left[\hat{H}\Omega - \Omega \hat{H}_{eff} \right] P_M^{(p,h)} = 0$$

Fig. 5.30: Division of the configuration space into model and outer space in effective Hamiltonian (EH) theory and into model, intermediate, and outer space in intermediate Hamiltonian (IH) theory. P and Q denote the respective projection operators.

When the model space is not energetically well separated from the outer space, this method faces convergence problems. This is commonly termed as the intruder state problem. In the intermediate Hamiltonian (IH) formulation, configuration space is divided into three subspaces, namely, the main(M), the intermediate(I), and the outer(O) space (see Fig. 5.30) with projection operators P_M, P_I and Q_O, respectively. The intermediate space acts as a buffer between the model and the outer space. When diagonalization the IH, a subset of the eigenvalues correspond to the main space obtained through EH theory. The IH is for the singly excited state sector (1,1) is defined as

$$H_I^{(1,1)} = P_O^{(1,1)} \bar{H} P_O^{(1,1)} + P_O^{(1,1)} \bar{H} Y^{(1,1)} P_M^{(1,1)}$$

where

$$Y^{(1,1)} = Q_O^{(1,1)} \left\{ S_2^{(0,1)} + S_2^{(1,0)} + S_2^{(0,1)} S_1^{(1,0)} + S_2^{(1,0)} S_1^{(0,1)} + S_2^{(1,0)} S_2^{(0,1)} \right\} P_M^{(1,1)}$$

The $S^{(1,0)}$ and $S^{(0,1)}$ are extracted from converged EOMIP-CCSD and EOMEA-CCSD calculations, respectively, by invoking intermediate normalization on the suitably chosen eigenvectors corresponding to active holes and active particles. The total procedure can be described as following

- solve the ground state coupled cluster equations
- construct $\hat{\bar{H}} = e^{-\hat{T}} \hat{H} e^{\hat{T}}$
- solve the EOMIP and EOMEA equations
- extract the $\hat{S}$ amplitudes

- construct the second similarity transformed Hamiltonian as $H_I^{(1,1)}$
- diagonalize the $H_I^{(1,1)}$ in CIS space

The automatic active space selection scheme and all the speed up options which are available for STEOM-CCSD, including bt-PNO and COSX, are also available for IH-FSMR-CCSD. All the keywords controlling the IH-FSMR-CCSD are similar to STEOM-CCSD as described in *Excited States via STEOM-CCSD*.

No UHF variant of IH-FSMR-CCSD is currently available.

5.11.2 Properties

The transition properties can be calculated using a simple CIS-like formulation, employing the converged IH-FSMR-CC eigenvectors. The transition moments are computed by default in an IH-FSMR-CCSD calculation.

5.11.3 Solvation Correction

Solvent effects can be approximated by a simple perturbative correction to the IH-FSMR-CCSD via

$$\omega_k = \omega_k^0 + \frac{1}{2}\bar{V}^{\Delta}\bar{Q}^{\Delta}$$

where

$$\omega_k^0 = \hat{L_K} H_I^{(1,1)} \hat{R_K}$$

The CPCM correction directly enters the $H_I^{(1,1)}$, the modified Hatree-Fock orbitals. In the non-equilibrium regime, one can simply write the perturbative correction as

$$\omega_k^{neq} = \omega_k^{0,neq} + \frac{1}{2}\bar{V}\left(P_{\Delta}^{neq}\right)\bar{Q}\left(P_{\Delta}^{neq}\right)$$

where

$$P_{\Delta}^{neq} = L_k R_k$$

A typical input file looks like

```
! aug-cc-pVDZ IH-FSMR-CCSD
!CPCM(water)
%mdci
NROOTS 8
DoSOLV true
DTol 1e-10
end
*xyz 0 1
O       0.0000  0.0000  0.1173
H       0.0000  0.7572 -0.4692
H       0.0000 -0.7572 -0.4692
*
```

For the above input, the following output is obtained:

```
---------------------------------
     CALCULATED SOLVENT SHIFTS
            CPCM  MODEL
---------------------------------

Contributions of the 'fast' term to the solvent shift

 State Shift(Eh) Shift(eV) Shift(cm**-1) Shift(nm) E_FSMRCC(eV)  E_FSMRCC+SHIFT(eV)
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------------------------------------------
   0: -0.0058000 -0.158    -1273.0        3.3      7.814          7.656
   1: -0.0133523 -0.363    -2930.5        5.0      9.650          9.287
   2: -0.0053622 -0.146    -1176.9        1.8     10.144          9.998
   3: -0.0078092 -0.212    -1713.9        2.2     10.958         10.746
   4: -0.0040294 -0.110     -884.4        1.0     11.534         11.424
   5: -0.0137147 -0.373    -3010.0        3.3     12.003         11.630
   6: -0.0093172 -0.254    -2044.9        2.2     12.131         11.878
   7: -0.0077514 -0.211    -1701.2        1.7     12.562         12.352
```

Thee perturbative correction only changes the transition energies and neither the wave function nor the transition moment.

5.12 Excited States via PNO-based coupled cluster

Despite the successes of the DLPNO-CC approximation for ground states, the use of PNOs for excited states has been less fruitful. Defining a PNO-based scheme for excited states that maintains the balance between speed and accuracy observed for ground states is not straightforward. As an intermediate solution, the basis for ground-state DLPNO quantities is transformed back to the canonical basis and used within the canonical EOM routine. This procedure is justified, as the main bottleneck of the EOM-CCSD or STEOM-CCSD methods lies in the ground-state calculation. Approximating the ground-state CCSD amplitudes with MP2 amplitudes is also possible, as done in the EOM-CCSD(2) approach. However, it is not reliable and can lead to large errors when the reference HF wave function does not provide a reasonable zeroth-order approximation to the ground-state wave function. Note that the back-transformed PNO scheme (bt-PNO) described here is available for both open-shell (UHF (QROs) or ROHF reference) and closed-shell (RHF reference) systems.

5.12.1 General Description

The back-transformation of the ground-state DLPNO-CCSD amplitudes to the virtual space involves three steps. The amplitudes in the PNO basis are first converted to the PAO basis, then to the atomic orbital (AO) basis, and finally to the canonical MO basis[640]. For example, in the closed-shell case, we have

$$d^{ij}_{\tilde{\mu}\tilde{a}_{ij}} T^{ij}_{\tilde{a}_{ij}\tilde{b}_{ij}} d^{ij}_{\tilde{b}_{ij}\tilde{\mu}} \Rightarrow L^{ij}_{\mu\tilde{\mu}} T^{ij}_{\tilde{\mu}\tilde{\nu}} L^{ij}_{\tilde{\nu}\nu} \Rightarrow C^{ij}_{a\mu} T^{ij}_{\mu\nu} C^{ij}_{\nu b} \Rightarrow T^{ij}_{ab},$$

$$d^{i}_{\tilde{\mu}\tilde{a}_{ii}} T^{i}_{\tilde{a}_{ii}} \Rightarrow L^{i}_{\mu\tilde{\mu}} T^{i}_{\tilde{\mu}} \Rightarrow C^{i}_{a\mu} T^{i}_{\mu} \Rightarrow T^{i}_{a},$$

The AO basis functions are denoted as $\mu, \nu, \ldots$, while $\tilde{\mu}, \tilde{\nu}, \ldots$ refers to PAOs. The missing pairs are treated using MP2 amplitudes. If all the thresholds are set to zero, the back-transformed amplitudes match exactly with the canonical RI-EOM-CCSD ones. On the other hand, when all the thresholds are made infinitely tight, one obtains the EOM-CCSD(2) results. This PNO-based excited state approach is available for all the flavors of EOM-CCSD and for STEOM-CCSD in both open- and closed-shell systems.

Below, we list all the parameters that influence the DLPNO-CCSD-based excited state calculations

```
%mdci
#bt-PNO-EOM and STEOM parameters - defaults displayed
DoEOMMP2 true         # MP2 correction for missing pairs
DoRECAN  true         # recanonicalization of the occupied
                      # orbitals before the excited state calculation
                      #(only relevant for the RHF implementation)
end
```

5.12.2 Reference State Energy

It should be noted that the reference energy for PNO-based EOM-CCSD or STEOM-CCSD differs slightly from the energy printed for a converged ground-state DLPNO-CCSD calculation, as it includes perturbative corrections for various truncated quantities.

```
----------------------
COUPLED CLUSTER ENERGY
----------------------

E(0)                                          ...   -113.913498239
E(CORR)(strong-pairs)                         ...     -0.401457078
E(CORR)(weak-pairs)                           ...     -0.000339627
E(CORR)(corrected)                            ...     -0.401796705
E(TOT)                                        ...   -114.315294944
```

In the bt-PNO-EOM-CCSD scheme, the CI-like excited state treatment of the reference state is defined by back-transformed DLPNO amplitudes (or MP2 amplitudes for the weak pairs). The energy corresponding to this set of amplitude is printed at the beginning of the EOM calculations.

```
Dressing integrals for EOM-CCSD             ...
Reference state energy for EOM-DLPNO-CCSD     ...   -114.314954945

done (    0.4)
```

Therefore, to calculate the total energy of an excited (ionized or electron attached) state, one needs to add the excitation energy to the reference state energy in bt-PNO-EOM-CCSD.

5.12.3 Use of Local Orbitals

The use of local orbitals makes it difficult to follow a particular guess vector during the Davidson diagonalization process in EOM-CC and STEOM-CC. Therefore, it is advisable to recanonicalize the occupied orbitals after the ground-state DLPNO-CCSD calculation by setting DoRECAN to true (this is only relevant for the closed-shell RHF implementation). It is important to note that recanonicalization does not affect the EOM-CCSD energies. However, STEOM-CC energies are not invariant to orbital rotations and may differ slightly between local and canonical orbitals. In the open-shell bt-PNO implementation, a different procedure is followed: all quantities are transformed to the delocalized basis before proceeding with the back-transformation and the excited-state calculation.

5.12.4 Some tips and tricks for bt-PNO calculations

- The bt-PNO scheme with tightPNO settings gives results, which are within 0.01 eV of the canonical EOM-CCSD numbers, at a fraction of the computational cost[668]. So, use of bt-PNO scheme is always preferable over canonical calculations.
- In the case of an RHF reference, one should set 'DLPNOLINEAR true' and 'NEWDOMAINS true' in the mdci block input to use the 2015 fully linear scaling implementation, which is more robust than the 2013 implementation used as default in bt-PNO scheme.
- The transition moment in bt-PNO-EOM (RHF only) and bt-PNO-STEOM (RHF, UHF (QROs) or ROHF) is only available using the linear approximation.

5.13 Excited States via DLPNO-STEOM-CCSD

The powerful coupled cluster methods can be used for excited states also. STEOM-CCSD is an improvement over the costly EOM-CCSD method which allows a lower computational scaling for calculating excitation energies. The DLPNO approximation on the orbital space makes the theory even more compact by further optimizing the computational cost. For general description, please read *Excited States via STEOM-CCSD*.

5.13.1 PNO dressing (experimental keyword)

In STEOM-DLPNO-CCSD method most of the steps are done using the powerful PNO approximation except the last one corresponding to the STEOM-CCSD calculation itself. In the canonical version of the dressing (default in ORCA) all the amplitudes previously computed at the PNO level are recanonicalized which increases the calculation cost.PNO dressing uses some of the PNO intermediates to reduce the dressing time. It has however no effect on the storage size.

Using this PNO dressing will reduce the calculation timing but at a reduce accuracy for the excitation energies. We thus recommend this option only for specific large systems (>2500 basis functions).

The keyword **DirectDressSTEOM** is set to true enable this option.

5.13.2 Keywords from STEOM-CCSD

Most of the keywords from STEOM-CCSD can be used within the DLPNO version. More information on the *Excited States via STEOM-CCSD* section.

5.13.3 Example

We would recommend this input for standard organic molecules. Formaldehyde is used as an example with def2-TZVP basis set.

As a general guideline we will discuss some of the keywords used there:

```
! STEOM-DLPNO-CCSD def2-TZVP def2-TZVP/C def2/J TightSCF

%mdci
  NRoots 6
  DoRootWise true
  OThresh 0.005
  VThresh 0.005
  TCutPNOSingles 1e-11
  NDAV 400
  DoStoreSTEOM true
  DoSimpleDens false
  AddL2Term True
  DTol 1e-5
end

* xyz 0 1
  C     0.016227   -0.000000    0.000000
  O     1.236847    0.000000   -0.000000
  H    -0.576537    0.951580   -0.000000
  H    -0.576537   -0.951580   -0.000000
*
```

5.13.4 Tips and Tricks

Important

For DLPNO calculations we need at least one auxiliary basis set. Therefore we have used def2-TZVP/C def2/J for this input. Without using explicit auxiliary basis sets, a generic keyword "AutoAux" can also be used.

In that case, the first line of the same input would look like

```
! STEOM-DLPNO-CCSD def2-TZVP AutoAux TightSCF
```

Note

DLPNO-STEOM-CCSD is not yet available for UHF wavefunctions. Therefore open shell calculations are currently not possible.

- TIGHTSCF is a must for any CCSD calculation.
- We will recommend using TIGHTPNO for all molecules as it is not a lot more expensive and helps achieving a better convergence.
- The OTHRESH, VTHRESH and TCutPNOSingles keywords help with converging the calculations, increasing the percentage active of each root. In contrary to standard STEOM-CCSD, we would acknowledge that the roots are converged when the percentage active character is at least 96%. Of course you have to check that the amplitude and orbitals associated with the excitation are correct (and what you expect). Tightening the 3 keywords mentioned above will increase this percentage active character. The most important being the TCutPNOSingles one. Be careful, the computational cost increases exponentially when thresholds are tightened. In most cases 10^{-12} or 5×10^{-12} should be enough. For OTHRESH and VTHRESH you should not go below 10^{-3}, as the benefits are not so obvious. A trick to achieve a better convergence is to play with the number of roots. It is often not necessary to compute a lot of roots if you are only interested in the first 3 for example. If some high energy roots have some low percentage active character, removing them can help for the convergence of other roots.
- As shown in the related paper[669], the STEOM density with the L2term is now the default for computing the excited state properties.
- The choice of the basis set can also speed up the calculation without a significant loss of accuracy. For most organic molecules def2-TZVP(-f) is enough. Trying def2-SVP is also a good idea for preliminary tests.
- The STEOM-CCSD excitation energies are very dependent from the starting geometry. Geometries optimised with several DFT functional can yield significant differences for the excitation energies (about 0.1 eV).
- It has happened that some roots were missing with the keyword **Dorootwise** set to true. Turning it off solved the issue but this keyword should be on by default.

5.14 One Photon Spectroscopy

5.14.1 General Description

Introduced in Orca 6.0, the One Photon Spectroscopy (OPS) tool takes charge of computing one-photon absorption (OPA), emission (OPE), and natural electric circular dichroism (ECD) intensities. In each case, the intensity of the spectrum ($I(\omega)$) resulting from a transition between an initial state I and a final state J is determined by the square modulus of the transition moment $T_{IJ} = \langle \Psi_I | \hat{H}_1 | \Psi_J \rangle$, weighted by the populations N_I and N_J of states I and J, respectively.

$$I(\omega) = \alpha\omega^{-1} \sum_{IJ} (N_I - N_J) |\langle \Psi_I | \hat{H}_1 | \Psi_J \rangle|^2 \delta(E_{JI} \pm \hbar\omega) \tag{5.84}$$

Where, α represents a positive constant, $\hbar\omega$ stands for photon energy, and the expression for $\hat{H}_1$ hinges on the specific modeling of photon-matter interaction.

5.14.2 Light-matter interaction approaches

Expressions for $\hat{H}_1$ can be derived from different theoretical perspectives. For instructional purposes, a classical electrodynamic approach is adopted here. By representing light classically through a vector potential ($\mathbf{A}$) and a scalar potential ($\phi = 0$), the radiation can be integrated into the Hamiltonian that models the molecular system.

$$\hat{H} = \sum_{i=1}^{N} \frac{1}{2m_e}\left[\hat{\mathbf{p}}_i - \frac{e}{c}\mathbf{A}(\mathbf{r}_i, t)\right]^2 - \frac{ge}{2m_e c}\sum_{i=1}^{N}\mathbf{B}(\mathbf{r}_i, t)\cdot\hat{s}_i + V(\mathbf{r}_1, ..., \mathbf{r}_N)$$

By disregarding the coupling between the magnetic field of light and the spin, as well as the $|\mathbf{A}|^2$ term, and applying Fermi's golden rule, an expression for $\langle\Psi_I|\hat{H}_1|\Psi_J\rangle$ under the Full-Semiclassical Light-Matter interaction can be derived.

$$T_{IJ}^{FFMIO} = \frac{e}{m_e}\sum_{i=1}^{N}\langle I|\,\mathcal{E}\cdot\left[e^{i\mathbf{k}\cdot\mathbf{r}_i}\hat{\mathbf{p}}_i\right]|J\rangle \tag{5.85}$$

Where T_{IJ}^{FFMIO} is the transition moment for the **F**ull (semiclassical) **F**ield-**M**atter **I**nteraction **O**perator (FFMIO), while k and $\mathcal{E}$ denote the wave and polarization vectors describing the light, respectively. $\mathbf{r}_i$ and $\mathbf{p}_i$ represent the position and linear momentum operators for the i-th electron.

Proceeding from eq. (5.85) and approximating the exponential term of T_{IJ}^{FFMIO} with a Taylor expansion yields various orders of interaction. The zeroth order ($T_{IJ}^{[0]}$) results in the electric dipolar velocity formulation ($ED\ vel$), which, upon molecular orientational averaging, yields:

$$\epsilon_{ED\ vel}(\omega) = \sum_{IJ}(N_I - N_J)\frac{2}{3E_{JI}}|\langle\Psi_I|\hat{p}|\Psi_J\rangle|^2\delta(E_{JI} \pm \hbar\omega) \tag{5.86}$$

For exact solutions of the Hamiltonian, or in theories that satisfy the Hypervirial theorem by construction, equation (5.86) can be transformed into a length formulation (5.87) ($ED\ len$).

$$\epsilon_{ED\ len}(\omega) = \sum_{IJ}(N_I - N_J)\frac{2E_{JI}}{3}|\langle\Psi_I|\hat{\mu}|\Psi_J\rangle|^2\delta(E_{JI} \pm \hbar\omega) \tag{5.87}$$

The absorption spectrum under $ED\ len$ and $ED\ vel$ formulations can be obtained in all Orca modules utilizing OPS by setting `DoDipoleLength` and `DoDipoleVelocity` to true, respectively, in the corresponding module's block. The results are then presented in the following tables:

```
-----------------------------------------------------------------------------------
↪-----------------
                    ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX      ␣
↪  DY        DZ
                      (eV)      (cm-1)     (nm)                 (au**2)    (au)     ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5   0.000000000   0.00000  -0.00000  ␣
↪0.00000   0.00000
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7   0.000000000   0.00000  -0.00000  ␣
↪0.00000  -0.00000
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5   0.000000000   0.00000   0.00000  ␣
↪0.00000  -0.00000
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3   0.301976959   2.47833   1.57427  -
↪0.00000  -0.00025
```

```
-----------------------------------------------------------------------------
↪-----------------
                      ABSORPTION SPECTRUM VIA TRANSITION VELOCITY DIPOLE MOMENTS
-----------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(P2)      P2        PX     ␣
↪  PY        PZ
                      (eV)      (cm-1)     (nm)                 (au**2)    (au)    ␣
↪ (au)      (au)
-----------------------------------------------------------------------------
↪-----------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5   0.000000000   0.00000   0.00000  ␣
↪0.00000  -0.00000
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7   0.000000000   0.00000   0.00000  ␣
↪0.00000  -0.00000
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5   0.000000001   0.00000  -0.00000  -
↪0.00000  -0.00001
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3   0.292852326   0.08029  -0.28335  ␣
↪0.00000  -0.00002
```

Here, transitions are denoted using spectroscopic notation, such as 1-3B3g representing 1^3B_{3g}. If symmetry is not specified in the calculation (or is unavailable in the selected method), the system defaults to C_1 point group. The **fosc(D2)** and **fosc(P2)** columns indicate the computed oscillator strengths in length and velocity formulations respectively. Additionally, **D2**, **P2**, **DX**, **DY**, **DZ**, **PX**, **PY**, and **PZ** represent the square modulus of the electric transition dipole moment, square modulus of the transition linear momentum, and its Cartesian components, respectively (with the imaginary unit in the linear momentum being implicit).

Alternatively, intensity computation directly using T_{IJ}^{FFMIO} is achievable in OPS through the utilization of the keyword `DoFullSemiclassical` set to true. In this scenario, the orientational average is computed semi-numerically due to the absence of available analytical expressions.

```
-----------------------------------------------------------------------------
                      ABSORPTION SPECTRUM VIA FULL SEMI-CLASSICAL FORMULATION
-----------------------------------------------------------------------------
     Transition      Energy     Energy  Wavelength   fosc(FFMIO)
                      (eV)      (cm-1)     (nm)
-----------------------------------------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000049901134
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000000000260
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000116403914
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.29285049449335
```

Contributions to the intensity can also be decomposed beyond the dipole approximation; however, origin dependence must be treated carefully and typically requires expert-level understanding of the theory. The first-order term ($T_{IJ}^{[1]}$) in the Taylor expansion of T_{IJ}^{FFMIO} gives rise to the electric quadrupole velocity formulation and the magnetic dipole contributions.

$$\langle\Psi_I|(T_{EQvel})_{\alpha\beta}|\Psi_J\rangle = \frac{ie}{2m_e}k_\alpha\epsilon_\beta\langle\Psi_I|r_\alpha p_\beta + p_\alpha r_\beta|\Psi_J\rangle \tag{5.88}$$

$$\langle\Psi_I|T_{MD}|\Psi_J\rangle = \frac{ie}{2m_e}\langle\Psi_I|(k\times\epsilon)(r\times p)|\Psi_J\rangle \tag{5.89}$$

Under the same conditions applied to the electric dipole transition moment, the quadrupole contribution can be reformulated in terms of a length representation.

$$\langle\Psi_I|(T_{EQlen})_{\alpha\beta}|\Psi_J\rangle = \frac{eE_{JI}}{2m_e}\sum_i k_\alpha\epsilon_\beta\langle\Psi_I|r_\alpha r_\beta|\Psi_J\rangle \tag{5.90}$$

By squaring the modulus of ($T_{IJ}^{[0]} + T_{IJ}^{[1]}$), six terms emerge in the oscillation strength: the dipole square contribution as listed previously in the dipole-approximation tables, a magnetic dipole square, an electric quadrupole square, and three cross-product terms: electric dipole-electric quadrupole, electric dipole-magnetic dipole, and magnetic dipole-electric quadrupole. Enabling `DecomposeFosc` to true allows for the estimation of spectrum intensity by including all three squared terms in the calculation of the intensity.

```
-----------------------------------------------------------------------------------
↪-----------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM
-----------------------------------------------------------------------------------
↪-----------------------------------------------
    Transition      Energy     Energy  Wavelength fosc(D2)  fosc(M2)  fosc(Q2)  ␣
↪fosc(D2+M2+Q2)     D2/TOT    M2/TOT    Q2/TOT
                     (eV)      (cm-1)     (nm)       (au)    (au*1e6)  (au*1e6)
-----------------------------------------------------------------------------------
↪-----------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000   0.49891   0.00010   0.
↪00000049901544   0.00000   0.99979   0.00021
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000   0.00000   0.00000   0.
↪00000000000000   0.00000   0.00000   0.00000
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000   1.12904   0.03298   0.
↪00000116202090   0.00000   0.97162   0.02838
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.30198   0.00000   0.00000   0.
↪30197695902797   1.00000   0.00000   0.00000
```

```
-----------------------------------------------------------------------------------
↪-----------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM (Velocity)
-----------------------------------------------------------------------------------
↪-----------------------------------------------
    Transition      Energy     Energy  Wavelength fosc(P2)  fosc(M2)  fosc(Q2)  ␣
↪fosc(P2+M2+Q2)     P2/TOT    M2/TOT    Q2/TOT
                     (eV)      (cm-1)     (nm)       (au)    (au*1e6)  (au*1e6)
-----------------------------------------------------------------------------------
↪-----------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000   0.49891   0.00010   0.
↪00000049901234   0.00000   0.99980   0.00020
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000   0.00000   0.00000   0.
↪00000000000006   0.99964   0.00012   0.00024
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000   1.12904   0.03441   0.
↪00000116403924   0.00051   0.96993   0.02956
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.29285   0.00000   0.00000   0.
↪29285232633663   1.00000   0.00000   0.00000
```

Here, the columns **fosc(D2+M2+Q2)** and **fosc(P2+M2+Q2)** refer to the total oscillator strengths obtained in length and velocity formulations, respectively, while the columns **fosc(D2)** (or **fosc(P2)**), **fosc(M2)**, and **fosc(Q2)** refer to the individual contributions of each term.

In scenarios where the dipolar contribution to the intensity is non-zero, dividing the total intensity into **D2** (or **P2**), **M2**, and **Q2** contributions becomes challenging due to the dependence of **M2** and **Q2** terms on the chosen origin. To address this origin-dependence issue, OPS offers additional formulations to the multipolar expansion. One possible approach devised by our group describes each transition from an origin that minimizes the contributions of **M2** and **Q2**, thereby redistributing the intensity from these terms to all other components in the expansion of $e^{i\mathbf{k}\cdot\mathbf{r}_i}$. This refined formulation is readily accessible in an "origin-adjusted" table. [670], [645]

```
-----------------------------------------------------------------------------------
↪-----------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM (origin adjusted)
-----------------------------------------------------------------------------------
↪-----------------------------------------------
    Transition      Energy     Energy  Wavelength fosc(D2)  fosc(M2)  fosc(Q2)  ␣
↪fosc(D2+M2+Q2)     D2/TOT    M2/TOT    Q2/TOT
                     (eV)      (cm-1)     (nm)       (au)    (au*1e6)  (au*1e6)
-----------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
↪---------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000   0.49891   0.00010   0.
↪00000049901544   0.00000   0.99979   0.00021
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000   0.00000   0.00000   0.
↪00000000000000   0.00000   0.00000   0.00000
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000   1.12904   0.03298   0.
↪00000116202090   0.00000   0.97162   0.02838
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.30198   0.00000   0.00000   0.
↪30197695902797   1.00000   0.00000   0.00000
```

Alternatively, the intensity described through a truncated expansion of $e^{i\mathbf{k}\cdot\mathbf{r}_i}$ be achieved in an origin-independent manner. In this scenario, the cross-terms between the electric dipole moment and the electric octupole, as well as between the electric dipole moment and the magnetic quadrupole moments, arise from the second-order expansion ($T^{[2]}_{IJ}$), resolving the origin dependence on the **D2**, **M2**, and **Q2** contributions. This formulation is provided in both length and velocity representations for the electric operators by setting the keyword `DecomposeFosc` to true, and the results are presented in the tables "Origin Independent, Length" and "Origin Independent, Velocity," respectively."

```
-----------------------------------------------------------------------------------
↪---------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM (Origin Independent, Length)
-----------------------------------------------------------------------------------
↪---------------------------------------------
     Transition      Energy     Energy  Wavelength fosc(D2)  fosc(M2)  fosc(Q2)␣
↪fosc(D2+M2+Q2+DM+DO) D2/TOT    M2/TOT    Q2/TOT
                      (eV)      (cm-1)    (nm)       (au)    (au*1e6)  (au*1e6)
-----------------------------------------------------------------------------------
↪---------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000   0.49891   0.00010   0.
↪00000049901544   0.00000   0.99979   0.00021
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000   0.00000   0.00000   0.
↪00000000000000   0.00000   0.00000   0.00000
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000   1.12904   0.03298   0.
↪00000116202090   0.00000   0.97162   0.02838
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.30198   0.00000   0.00000   0.
↪30197506344159   1.00001   0.00000   0.00000
```

```
-----------------------------------------------------------------------------------
↪---------------------------------------------
                     ABSORPTION SPECTRUM COMBINED ELECTRIC DIPOLE + MAGNETIC␣
↪DIPOLE + ELECTRIC QUADRUPOLE SPECTRUM (Origin Independent, Velocity)
-----------------------------------------------------------------------------------
↪---------------------------------------------
     Transition      Energy     Energy  Wavelength fosc(P2)  fosc(M2)  fosc(Q2)␣
↪fosc(P2+M2+Q2+PM+PO) P2/TOT    M2/TOT    Q2/TOT
                      (eV)      (cm-1)    (nm)       (au)    (au*1e6)  (au*1e6)
-----------------------------------------------------------------------------------
↪---------------------------------------------
  0-1Ag ->  1-1B3g  2.507275   20222.5   494.5     0.00000   0.49891   0.00010   0.
↪00000049901234   0.00000   0.99980   0.00020
  0-1Ag ->  2-1Au   2.726731   21992.6   454.7     0.00000   0.00000   0.00000   0.
↪00000000000006   0.99962   0.00012   0.00024
  0-1Ag ->  3-1B2g  3.892633   31396.2   318.5     0.00000   1.12904   0.03441   0.
↪00000116403924   0.00051   0.96993   0.02956
  0-1Ag ->  4-1B3u  4.973431   40113.4   249.3     0.29285   0.00000   0.00000   0.
↪29285049443551   1.00001   0.00000   0.00000
```

The tables presented in this section can be generated using the following input example:

```
!B3LYP def2-TZVPP UseSym
```

(continues on next page)

(continued from previous page)

```
%sym SymThresh 1.0e-2 end

%CIS
  Nroots 4
  Tda false
  DoDipoleLength true
  DoDipoleVelocity true
  DecomposeFosc true
  DoFullSemiclassical true
END

* xyz 0 1
  C  -0.03361342468929      0.00165373985356     -0.01949461738929
  C   0.00156939445126      0.00011912087577      1.46272959096597
  C   1.15884465742251      0.00010247651897      2.13104196345420
  C   2.45992795249087      0.00152015393852      1.42032876326693
  C   2.42474592208737      0.00010359895701     -0.06189678799337
  C   1.26746963770198      0.00012121580349     -0.73020920027406
  H  -0.96132586605922     -0.00069823088896      1.95925560299438
  H   1.20967888389840     -0.00071666462249      3.21322810501222
  O   3.51487697965937     -0.00034474583300      2.02878006087658
  H   3.38764113128475     -0.00071842966522     -0.55842278191419
  H   1.21663501697728     -0.00069837665390     -1.81239534279079
  O  -1.08856228522528     -0.00044385828375     -0.62794535620861
*
```

5.14.3 Natural electric circular dichroism

In the case of ECD modeling, intensity is determined by computing the difference between two absorption spectra: one acquired using left-circularly polarized light and the other utilizing right-circularly polarized light.

$$\Delta I(\omega) = I_{left}(\omega) - I_{rigth}(\omega)$$

To initiate these calculations, the keyword `DoCD` should be set to true. There are three available implementations, which involve utilizing the length or velocity representations for the electric dipole moment, and also using the FFMIO, selected by including the keywords `DoDipoleLength`, `DoDipoleVelocity`, and `DoFullSemiclassical` respectively.

```
-----------------------------------------------------------------------------------
↪-------
                      CD SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-------
     Transition      Energy     Energy  Wavelength    R        MX        MY       ␣
↪MZ
                      (eV)      (cm-1)     (nm)    (1e40*cgs)   (au)      (au)     ␣
↪(au)
-----------------------------------------------------------------------------------
↪-------
  0-1A  ->  1-1A    5.858500   47252.0   211.6   -9.15001  -0.01149   0.10351  -0.
↪06803
  0-1A  ->  2-1A    6.859736   55327.5   180.7  -10.82714   0.13087   0.17869   0.
↪25077
  0-1A  ->  3-1A    7.025193   56662.0   176.5   10.05878  -0.00526  -0.09041  -0.
↪13618
  0-1A  ->  4-1A    7.837041   63210.0   158.2   33.15402   0.02541  -0.23067   0.
↪15096
```

```
-------------------------------------------------------------------------------
↪-------
                      CD SPECTRUM VIA TRANSITION VELOCITY DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪-------
     Transition      Energy     Energy  Wavelength    R        MX        MY       ␣
↪MZ
                      (eV)      (cm-1)     (nm)    (1e40*cgs)   (au)      (au)     ␣
↪(au)
-------------------------------------------------------------------------------
↪-------
  0-1A  ->  1-1A    5.858500   47252.0   211.6  -10.85364  -0.01149   0.10351  -0.
↪06803
  0-1A  ->  2-1A    6.859736   55327.5   180.7  -15.54410   0.13087   0.17869   0.
↪25077
  0-1A  ->  3-1A    7.025193   56662.0   176.5   12.44178  -0.00526  -0.09041  -0.
↪13618
  0-1A  ->  4-1A    7.837041   63210.0   158.2   38.90662   0.02541  -0.23067   0.
↪15096
```

```
-------------------------------------------------------------------
                      CD SPECTRUM VIA FULL SEMI-CLASSICAL FORMULATION
-------------------------------------------------------------------
     Transition      Energy     Energy  Wavelength    R
                      (eV)      (cm-1)     (nm)    (1e40*cgs)
-------------------------------------------------------------------
  0-1A  ->  1-1A    5.858500   47252.0   211.6   -10.85394
  0-1A  ->  2-1A    6.859736   55327.5   180.7   -15.54405
  0-1A  ->  3-1A    7.025193   56662.0   176.5    12.44211
  0-1A  ->  4-1A    7.837041   63210.0   158.2    38.90630
```

The following input example may be used to generate the CD tables presented in this section:

```
!B3LYP def2-QZVPP

%CIS
  Nroots 4
  Tda false
  Docd true
  DoDipoleLength true
  DoDipoleVelocity true
  DoFullSemiclassical true
END

* xyz 0 1
  O   0.00292203187063     -0.00094788357265      0.00607763745451
  H  -0.01058222980812      0.00702286262037      0.97110447138368
  O   1.43049571483128     -0.00638910838156     -0.24427444074129
  H   1.54056670040621     -0.87944353346616     -0.64068822569690
*
```

5.14.4 SOC and SSC corrected spectrum

All the OPS keywords and tables listed above are available for the cases in which spin-orbit coupling and spin-spin coupling effects are taken into account by a QDPT formulation (when the method is available in the selected ORCA module). In those cases, ORCA 6.0 does not provide symmetry after the relativistic correction; therefore, C_1 group symmetry transitions are reported. Additionally, the multiplicity of the relativistically corrected roots are not well defined anymore; therefore, the average value is reported.

For the last case example presented in this section, when `DoSOC` is set to true in the input, OPS reports:

```
-----------------------------------------------------------------------------------
↪---------------------
     SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪---------------------
     Transition          Energy      Energy  Wavelength fosc(D2)       D2        |DX|␣
↪      |DY|       |DZ|
                          (eV)       (cm-1)     (nm)                 (au**2)     (au)␣
↪      (au)       (au)
-----------------------------------------------------------------------------------
↪---------------------
  0-1.0A  ->  1-3.0A    4.620455   37266.5   268.3   0.000000105   0.00000   0.
↪00079   0.00024   0.00049
  0-1.0A  ->  2-3.0A    4.620460   37266.5   268.3   0.000000009   0.00000   0.
↪00003   0.00024   0.00016
  0-1.0A  ->  3-3.0A    4.620491   37266.8   268.3   0.000000126   0.00000   0.
↪00093   0.00034   0.00035
  0-1.0A  ->  4-3.0A    5.626046   45377.1   220.4   0.000003140   0.00002   0.
↪00043   0.00397   0.00261
  0-1.0A  ->  5-3.0A    5.626322   45379.4   220.4   0.000000070   0.00000   0.
↪00056   0.00020   0.00040
  0-1.0A  ->  6-3.0A    5.626327   45379.4   220.4   0.000000003   0.00000   0.
↪00002   0.00013   0.00009
  0-1.0A  ->  7-1.0A    5.858722   47253.8   211.6   0.003490904   0.02432   0.
↪01409   0.12986   0.08520
  0-1.0A  ->  8-3.0A    6.537246   52726.4   189.7   0.000000006   0.00000   0.
↪00017   0.00005   0.00008
  0-1.0A  ->  9-3.0A    6.537246   52726.5   189.7   0.000000225   0.00000   0.
↪00011   0.00099   0.00065
  0-1.0A  -> 10-3.0A    6.537257   52726.5   189.7   0.000000032   0.00000   0.
↪00004   0.00037   0.00025
  0-1.0A  -> 11-1.0A    6.859693   55327.2   180.7   0.002900797   0.01726   0.
↪12934   0.02129   0.00891
  0-1.0A  -> 12-1.0A    7.025210   56662.1   176.5   0.011980549   0.06961   0.
↪23240   0.08590   0.09066
  0-1.0A  -> 13-3.0A    7.448215   60073.9   166.5   0.000000040   0.00000   0.
↪00018   0.00022   0.00037
  0-1.0A  -> 14-3.0A    7.448216   60073.9   166.5   0.000000003   0.00000   0.
↪00002   0.00011   0.00006
  0-1.0A  -> 15-3.0A    7.448231   60074.0   166.5   0.000000533   0.00000   0.
↪00116   0.00078   0.00098
  0-1.0A  -> 16-1.0A    7.837071   63210.2   158.2   0.012389093   0.06452   0.
↪02426   0.21144   0.13867
```

In this case, due to the complex nature of the relativistically-corrected wave functions, dipole moments are not necessarily real values, and their modulus are reported in the "|DX|", "|DY|", and "|DZ|" columns. Real and imaginary components may be obtained in extended tables by increasing the selected print level. In this instance, selecting `Printlevel 4` in the %CIS block.

```
-----------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪---------
```

(continues on next page)

(continued from previous page)

```
     SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS (-␣
↪Extended -)
-----------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪---------
     Transition          Energy     Energy  Wavelength     fosc(D2)          D2   ␣
↪       DX(re)         DX(im)         DY(re)         DY(im)         DZ(re)       ␣
↪ DZ(im)
                          (eV)      (cm-1)     (nm)                          (au**2)␣
↪        (au)           (au)           (au)           (au)           (au)        ␣
↪   (au)
-----------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪---------
  0-1.0A  ->  1-3.0A    4.620455   37266.5   268.3         1.05e-07       9.31e-07␣
↪      4.10e-04       6.80e-04      -1.22e-04      -2.02e-04      -2.55e-04      -
↪4.24e-04
  0-1.0A  ->  2-3.0A    4.620460   37266.5   268.3         9.18e-09       8.11e-08␣
↪     -2.25e-05       1.30e-05       2.05e-04      -1.18e-04      -1.35e-04      ␣
↪7.78e-05
  0-1.0A  ->  3-3.0A    4.620491   37266.8   268.3         1.26e-07       1.11e-06␣
↪      4.17e-04      -8.35e-04       1.51e-04      -3.01e-04       1.58e-04      -
↪3.17e-04
  0-1.0A  ->  4-3.0A    5.626046   45377.1   220.4         3.14e-06       2.28e-05␣
↪     -4.31e-04       5.49e-07       3.97e-03      -5.06e-06      -2.61e-03      ␣
↪3.32e-06
  0-1.0A  ->  5-3.0A    5.626322   45379.4   220.4         7.04e-08       5.11e-07␣
↪      1.13e-04       5.45e-04      -4.10e-05      -1.98e-04      -8.13e-05      -
↪3.92e-04
  0-1.0A  ->  6-3.0A    5.626327   45379.4   220.4         3.42e-09       2.48e-08␣
↪     -1.52e-05      -1.15e-07       1.31e-04       9.97e-07      -8.58e-05      -
↪6.51e-07
  0-1.0A  ->  7-1.0A    5.858722   47253.8   211.6         3.49e-03       2.43e-02␣
↪     -1.41e-02      -2.11e-05       1.30e-01       1.94e-04      -8.52e-02      -
↪1.28e-04
  0-1.0A  ->  8-3.0A    6.537246   52726.4   189.7         6.35e-09       3.97e-08␣
↪     -1.24e-04       1.22e-04       3.61e-05      -3.56e-05       5.89e-05      -
↪5.80e-05
  0-1.0A  ->  9-3.0A    6.537246   52726.5   189.7         2.25e-07       1.40e-06␣
↪      1.10e-04       1.07e-06      -9.87e-04      -9.61e-06       6.46e-04      ␣
↪6.29e-06
  0-1.0A  -> 10-3.0A    6.537257   52726.5   189.7         3.24e-08       2.02e-07␣
↪     -4.25e-05      -5.39e-08       3.73e-04       4.74e-07      -2.47e-04      -
↪3.13e-07
  0-1.0A  -> 11-1.0A    6.859693   55327.2   180.7         2.90e-03       1.73e-02␣
↪      4.49e-02       1.21e-01       7.39e-03       2.00e-02       3.09e-03      ␣
↪8.35e-03
  0-1.0A  -> 12-1.0A    7.025210   56662.1   176.5         1.20e-02       6.96e-02␣
↪     -2.18e-01       8.17e-02      -8.04e-02       3.02e-02      -8.49e-02      ␣
↪3.19e-02
  0-1.0A  -> 13-3.0A    7.448215   60073.9   166.5         3.95e-08       2.17e-07␣
↪      6.65e-05      -1.67e-04      -8.19e-05       2.05e-04      -1.37e-04      ␣
↪3.42e-04
  0-1.0A  -> 14-3.0A    7.448216   60073.9   166.5         2.93e-09       1.61e-08␣
↪     -2.09e-05       6.86e-07       1.12e-04      -3.68e-06      -5.57e-05      ␣
↪1.83e-06
  0-1.0A  -> 15-3.0A    7.448231   60074.0   166.5         5.33e-07       2.92e-06␣
↪     -6.54e-04      -9.63e-04      -4.40e-04      -6.48e-04      -5.48e-04      -
↪8.07e-04
  0-1.0A  -> 16-1.0A    7.837071   63210.2   158.2         1.24e-02       6.45e-02␣
↪     -2.40e-02       3.67e-03       2.09e-01      -3.20e-02      -1.37e-01      ␣
```

(continues on next page)

(continued from previous page)

```
↪2.10e-02
```

5.14.5 OPS Full list of keywords

The folowing list of keywords may be included directly in the correcponding module block to trigger OPS compute the corresponding intensities.

```
DoCD                    (to request circular dichroism calculation)
DoDipoleLength          (to request the use of electric moments in a length␣
↪formulation)
DoDipoleVelocity        (to request the use of electric moments in a velocity␣
↪formulation)
DoFullSemiclassical     (to request the calculation of complete semiclassical␣
↪multipolar moments)
DecomposeFosc           (to request the decomposition of the oscillator strengths in␣
↪a multipolar expansion)
```

In Orca 6.1, the modules in which OPS is available are: `%cis`, `%tddft`, `%rocis`, `%casscf`, `%mrci`, `%mdci`, `%lft`, `%mcrpa`, `%rasci` and `%xes`.

5.14.6 Notes

1. All values in the OPS tables are expressed in atomic units (au) unless otherwise specified.
2. Keyword corrections are done internally when necessary.
3. **IMPORTANT**: Input keywords have been standardized across all OPS-utilizing modules; Orca 5.x are no longer valid.
4. **IMPORTANT**: In CASSCF and MRCI, keywords controlling the absorption and ECD spectrum no longer belong in the `%rel` block.
5. Origin-adjusted formulation is approximated in SOC/SSC formulations.
6. OPS is not available for sTDDFT and sTDADFT.
7. The MDCI module reports spectra using left, right, and both solutions.
8. ADC2 and EOM-CCSD methods exclusively utilize ED length tables.
9. Starting in ORCA 6.1, the keywords `DoHigherMoments`, `DecomposeFoscLength` and `DecomposeFoscVelocity` have been consolidated into a single, simplified keyword: `DecomposeFosc`.

5.15 Core-Level Spectroscopy with Coupled Cluster Methods

The equation of motion coupled cluster method and its similarity transformed version provides an easy way to directly calculate core-ionization and core-excitation energies. The core-level spectroscopy with EOM-CCSD is only available for closed shell systems.

5.15.1 Core-Ionization

One can obtain core-ionized states if one calculates a large no of roots. The ORCA implementation of IP-EOM-CCSD, however, allows one to directly target the ionization from the core-orbitals. A typical IP-EOM-CCSD input file for the acetic acid will look like

```
!IP-EOM-CCSD ExtremeSCF cc-pvdz
!NoFrozencore
%maxcore 5000

%mdci
nroots 4        # no of roots
FollowCIS true  # Follow  the initial guess orbital
DoCVS true      # Core valence separation (currently both the option needs to be␣
↪true)
DoCore true     # Directly target the core
CoreOrb[0]=0    # The MO from which it will start counting
maxiter 500     # no of iteration, generally requires larger no of roots
end

*xyz 0 1
 C             -6.7624010562   0.1328615492   0.0389382700
 C             -5.3564667033   0.2819965475  -0.5188248498
 H             -6.9983743824   1.0019615710   0.6510029634
 H             -7.4924880320   0.0542210905  -0.7741766747
 H             -6.8380664832  -0.7720291637   0.6519904379
 O             -4.9303467983  -0.7518088469  -1.3223158759
 H             -5.6257914271  -1.4265892921  -1.4015111180
 O             -4.6208051175   1.2132365445  -0.3081931529
*
```

The output of it will be

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: 19.999934 au   544.226 eV 4389478.1 cm**-1
  Amplitude    Excitation
   0.672786     0 -> x
Percentage singles character=     82.70

IROOT=  2: 19.944006 au   542.704 eV 4377203.3 cm**-1
  Amplitude    Excitation
   0.672143     1 -> x
Percentage singles character=     82.41

IROOT=  3: 10.962172 au   298.296 eV 2405918.7 cm**-1
  Amplitude    Excitation
   0.669804     2 -> x
Percentage singles character=     81.37

IROOT=  4: 10.826519 au   294.605 eV 2376146.3 cm**-1
  Amplitude    Excitation
   0.670634     3 -> x
Percentage singles character=     81.74
```

The option 'DoCore true' together with 'DoCVS true' and 'CoreOrb[0]=0' is enough to enable the calculation for core ionizations in IP-EOM-CCSD. The option 'CoreOrb[0]=0' indicates that we will look for core excitations arising from the orbital 0 up to the HOMO, which correspond to the oxygen K-shell. We can also directly access the carbon K-edge by starting from the orbital 2 instead. For this, we use 'CoreOrb[0]=2'

```
!IP-EOM-CCSD ExtremeSCF cc-pvdz
!NoFrozencore
%maxcore 5000

%mdci
nroots 2        # no of roots
FollowCIS true  # Follow  the initial guess orbital
DoCVS true      # Core valence separation (currently both the option needs to be␣
↪true)
DoCore true     # Directly target the core
CoreOrb[0]=2    # The MO from which it will start counting
maxiter 500     # no of iteration, generally requires larger no of roots
end


*xyz 0 1
 C             -6.7624010562   0.1328615492   0.0389382700
 C             -5.3564667033   0.2819965475  -0.5188248498
 H             -6.9983743824   1.0019615710   0.6510029634
 H             -7.4924880320   0.0542210905  -0.7741766747
 H             -6.8380664832  -0.7720291637   0.6519904379
 O             -4.9303467983  -0.7518088469  -1.3223158759
 H             -5.6257914271  -1.4265892921  -1.4015111180
 O             -4.6208051175   1.2132365445  -0.3081931529
*
```

The output of it will be

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: 10.962172 au   298.296 eV 2405918.7 cm**-1
  Amplitude    Excitation
   0.669804     2 -> x
Percentage singles character=     81.37

IROOT=  2: 10.826519 au   294.605 eV 2376146.3 cm**-1
  Amplitude    Excitation
   0.670634     3 -> x
Percentage singles character=     81.74
```

Now, the core-ionized states remains embedded in the high density of doubly ionized valence states that form the continuum. This leads to severe convergence problems. One easy way to overcome this is to use the core-valence separation approximation which is turned on by setting an upper limit to the possible core excitations. The orbitals from which the contributions are not neglected for the core-valence separation are set by 'CoreOrb[0]= initial,final'. It is generally a good idea to include all the core orbitals corresponding to a particular element if one is interested in the ionization from any of the core orbitals for the particular element. In the second example both the carbon core-orbitals are also included, which is equivalent to set 'CoreOrb[0] = 0,3'. A 'bt-PNO-IP-EOM-CCSD' input file for the same example will look like

```
!bt-PNO-IP-EOM-CCSD ExtremeSCF cc-pvdz cc-pvdz/c
!NoFrozencore
%maxcore 5000

%mdci
nroots 4        # no of roots
FollowCIS true  # Follow  the initial guess orbital
DoCVS true      # Core valence separation (currently both the option needs to be␣
↪true)
DoCore true     # Directly target the core
```

(continues on next page)

(continued from previous page)

```
CoreOrb[0]=0,3  # The MO from which it will start counting
maxiter 500     # no of iteration, generally requires larger no of roots
end


*xyz 0 1
 C             -6.7624010562   0.1328615492   0.0389382700
 C             -5.3564667033   0.2819965475  -0.5188248498
 H             -6.9983743824   1.0019615710   0.6510029634
 H             -7.4924880320   0.0542210905  -0.7741766747
 H             -6.8380664832  -0.7720291637   0.6519904379
 O             -4.9303467983  -0.7518088469  -1.3223158759
 H             -5.6257914271  -1.4265892921  -1.4015111180
 O             -4.6208051175   1.2132365445  -0.3081931529
*
```

The output of it will be

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: 19.999730 au   544.220 eV 4389433.3 cm**-1
  Amplitude    Excitation
   0.672774     0 -> x
Percentage singles character=     82.69

IROOT=  2: 19.943800 au   542.698 eV 4377158.2 cm**-1
  Amplitude    Excitation
   0.672139     1 -> x
Percentage singles character=     82.41

IROOT=  3: 10.962316 au   298.300 eV 2405950.3 cm**-1
  Amplitude    Excitation
   0.669787     2 -> x
Percentage singles character=     81.36

IROOT=  4: 10.826562 au   294.606 eV 2376155.7 cm**-1
  Amplitude    Excitation
   0.670612     3 -> x
Percentage singles character=     81.73
```

The results are in excellent agreement with the canonical one. A DLPNO variant for the same example will look like

```
!IP-EOM-DLPNO-CCSD ExtremeSCF cc-pvdz cc-pvdz/c
!NoFrozencore
%maxcore 5000

%mdci
nroots 4        # no of roots
FollowCIS true  # Follow  the initial guess orbital
DoCVS true      # Core valence separation (currently both the option needs to be␣
↪true)
DoCore true     # Directly target the core
CoreOrb[0]=0,3  # The MO from which it will start counting
maxiter 500     # no of iteration, generally requires larger no of roots
end

*xyz 0 1
 C             -6.7624010562   0.1328615492   0.0389382700
 C             -5.3564667033   0.2819965475  -0.5188248498
```

(continues on next page)

(continued from previous page)

```
 H             -6.9983743824   1.0019615710   0.6510029634
 H             -7.4924880320   0.0542210905  -0.7741766747
 H             -6.8380664832  -0.7720291637   0.6519904379
 O             -4.9303467983  -0.7518088469  -1.3223158759
 H             -5.6257914271  -1.4265892921  -1.4015111180
 O             -4.6208051175   1.2132365445  -0.3081931529
*
```

The output of it will be

```
----------------------
EOM-CCSD RESULTS (RHS)
----------------------

IROOT=  1: 20.051396 au   545.626 eV 4400772.7 cm**-1
  Amplitude    Excitation
  -0.678316     0 -> x
Percentage singles character=    101.08

IROOT=  2: 19.986869 au   543.870 eV 4386610.6 cm**-1
  Amplitude    Excitation
  -0.677491     1 -> x
Percentage singles character=    101.14

IROOT=  3: 11.034622 au   300.267 eV 2421819.6 cm**-1
  Amplitude    Excitation
  -0.674692     2 -> x
  -0.108213     1 -> x    13 ->  16
  -0.108195     1 -> x    15 ->  16
Percentage singles character=    101.21

IROOT=  4: 10.890773 au   296.353 eV 2390248.3 cm**-1
  Amplitude    Excitation
   0.679269     3 -> x
Percentage singles character=    101.04
```

Although the error in the absolute IP values are as large as 1 eV, the so-called 'chemical shift' i.e. the difference between the IP value of two different atoms of the same elements are reasonably correct.

5.15.2 Core-Ionization (UHF)

Starting from ORCA 6.0.1 it is also possible to perform core-ionized states on open-shell systems.[671] This is a direct extension of the closed-shell implementation and the syntax is kept almost identical to that of closed-shell. Now, we need to define 'CoreOrb' for each one of the spin channels using 'CoreOrb[0]= min,max' for spin up and 'CoreOrb[1]=min,max' for spin down electrons.

```
   CoreOrb[0]= min,max        #The MO from which it will start counting (spin up)
   CoreOrb[1]= min,max        #The MO from which it will start counting (spin down)
```

In the following example, we perform a core-ionization on the dyoxygen molecule on its triplet ground-state for 'DoAlpha' and 'DoBeta' using ORCA's compound.

```
%compound

new_step

  !UHF IP-EOM-CCSD cc-pVDZ ExtremeSCF KDIIS nofrozencore

  %mdci
```

(continues on next page)

(continued from previous page)

```
  DoAlpha true
  NRoots 4
  DTol 1e-6
  DoCore true
  DoCVS  true
  CoreOrb[0]=0,1
  CoreOrb[1]=0,1
  end

  *xyz 0 3
    O      0.0 0.0 0.0
    O      0.0 0.0 1.207
  *
step_end

new_step
  !UHF IP-EOM-CCSD cc-pVDZ ExtremeSCF KDIIS nofrozencore

  %mdci
  DoBeta true
  NRoots 4
  DTol 1e-6
  DoCore true
  DoCVS  true
  CoreOrb[0]=0,1
  CoreOrb[1]=0,1
  end

  *xyz 0 3
    O      0.0 0.0 0.0
    O      0.0 0.0 1.207
  *
step_end

end
```

The output for 'DoAlpha' looks like

```
-------------------------------
UHF IP-EOM-CCSD RESULTS (RHS)
-------------------------------

IROOT=  1: 20.137719 au   547.975 eV 4419718.6 cm**-1
  Amplitude    Excitation
   0.894649     0a -> x
   0.115737     0a -> x     5b ->  12b
   0.115737     0a -> x     6b ->  13b
  -0.123325     1a -> x     5b ->   7b
  -0.120347     1a -> x     5b ->   8b
  -0.120347     1a -> x     6b ->   7b
   0.123325     1a -> x     6b ->   8b
Percentage singles character=     80.04

IROOT=  2: 20.137008 au   547.956 eV 4419562.4 cm**-1
  Amplitude    Excitation
   0.894673     1a -> x
  -0.123144     0a -> x     5b ->   7b
  -0.120171     0a -> x     5b ->   8b
  -0.120171     0a -> x     6b ->   7b
   0.123144     0a -> x     6b ->   8b
   0.115678     1a -> x     5b ->  12b
```

(continues on next page)

(continued from previous page)

```
   0.115678     1a -> x     6b ->  13b
Percentage singles character=     80.04

IROOT=  3:  1.418592 au    38.602 eV  311344.9 cm**-1
  Amplitude    Excitation
   0.531451     2a -> x
   0.266889     4a -> x     3b ->   7b
   0.431978     4a -> x     3b ->   8b
   0.116970     4a -> x     3b ->  14b
  -0.431978     5a -> x     3b ->   7b
   0.266889     5a -> x     3b ->   8b
  -0.116970     5a -> x     3b ->  15b
  -0.112676     7a -> x     2b ->   8b
   0.105473     7a -> x     4b ->   8b
  -0.112676     8a -> x     2b ->   7b
   0.105473     8a -> x     4b ->   7b
Percentage singles character=     29.23

IROOT=  4:  0.964260 au    26.239 eV  211630.5 cm**-1
  Amplitude    Excitation
   0.825207     3a -> x
   0.147947     6a -> x     5b ->   7b
   0.144375     6a -> x     5b ->   8b
   0.144375     6a -> x     6b ->   7b
  -0.147947     6a -> x     6b ->   8b
  -0.267515     7a -> x     3b ->   8b
  -0.267515     8a -> x     3b ->   7b
Percentage singles character=     68.10
```

and for 'DoBeta'

```
-------------------------------
UHF IP-EOM-CCSD RESULTS (RHS)
-------------------------------

IROOT=  1: 20.102122 au   547.007 eV 4411905.8 cm**-1
  Amplitude    Excitation
   0.904712     0b -> x
Percentage singles character=     81.85

IROOT=  2: 20.100495 au   546.962 eV 4411548.8 cm**-1
  Amplitude    Excitation
   0.904679     1b -> x
Percentage singles character=     81.84

IROOT=  3:  1.482975 au    40.354 eV  325475.4 cm**-1
  Amplitude    Excitation
   0.911468     2b -> x
  -0.437544     2b -> x     3b ->  10b
   0.117502     2b -> x     3b ->  17b
  -0.181512     2b -> x     3b ->  27b
  -0.150802     2b -> x     4b ->  16b
   0.155724     3b -> x     4b ->   9b
  -0.362017     3b -> x     4b ->  10b
   0.170790     3b -> x     4b ->  27b
   0.414703     2b -> x     5b ->  18b
  -0.477670     3b -> x     5b ->  15b
   0.121834     4b -> x     5b ->  13b
  -0.266578     4b -> x     5b ->  19b
  -0.414703     2b -> x     6b ->  19b
   0.477670     3b -> x     6b ->  14b
```

(continues on next page)

(continued from previous page)

```
   0.121834     4b -> x     6b ->  12b
  -0.266578     4b -> x     6b ->  18b
  -0.132721     5b -> x     6b ->  20b
   0.918380     5b -> x     6b ->  21b
   0.170340     3b -> x     6a ->   9a
  -0.124771     5b -> x     7a ->   9a
  -0.124771     6b -> x     8a ->   9a
Percentage singles character=     83.16

IROOT=  4:  0.889071 au    24.193 eV  195128.5 cm**-1
  Amplitude    Excitation
   0.946265     3b -> x
   0.272057     4b -> x     5b ->   7b
  -0.223370     4b -> x     5b ->   8b
   0.223370     4b -> x     6b ->   7b
   0.272057     4b -> x     6b ->   8b
Percentage singles character=     89.54
```

5.15.3 Core-Excitation

The STEOM-CCSD approach provides an efficient and accurate way to do the K-edge core-excitation spectroscopy. A typical input file for the STEOM-CCSD will look like

```
!STEOM-CCSD ExtremeSCF cc-pCVTZ Bohrs NoFrozencore

%mdci
nroots 5
FollowCIS true
DoSimpleDens False # use exact STEOM transition moment
maxiter 500
DoCVS true
DoCore true
CoreOrb[0]=0,0
end

*xyz 0 1
O 0 0 0.913973
C 0 0 -1.218243
*
```

The output will be

```
-------------------------------
STEOM-CCSD RESULTS (SINGLETS)
-------------------------------

IROOT=  1: 19.685582 au   535.672 eV 4320485.8 cm**-1
  Amplitude    Excitation
  -0.932683     6 ->   8
   0.281517     6 ->  12
  -0.211943     6 ->  14
  Ground state amplitude:  0.000000

Percentage Active Character     99.41

  Amplitude    Excitation in Canonical Basis
  -0.846006     0 ->   7
  -0.428151     0 ->   8
   0.252869     0 ->  12
```

(continues on next page)

(continued from previous page)

```
   0.151677     0 ->  15

IROOT=  2: 19.685582 au   535.672 eV 4320485.9 cm**-1
  Amplitude    Excitation
  -0.932691     6 ->   7
   0.281484     6 ->  11
   0.211959     6 ->  15
  Ground state amplitude:  0.000000

Percentage Active Character     99.41

  Amplitude    Excitation in Canonical Basis
  -0.428151     0 ->   7
   0.846006     0 ->   8
  -0.252868     0 ->  11
  -0.151677     0 ->  14

IROOT=  3: 19.964529 au   543.262 eV 4381707.6 cm**-1
  Amplitude    Excitation
   0.978863     6 ->   9
   0.157708     6 ->  10
  -0.101384     6 ->  16
  Ground state amplitude: -0.001155

Percentage Active Character     99.39

  Amplitude    Excitation in Canonical Basis
  -0.978751     0 ->   9
   0.125406     0 ->  10
  -0.112036     0 ->  18
   0.102538     0 ->  21

IROOT=  4: 20.091389 au   546.714 eV 4409550.1 cm**-1
  Amplitude    Excitation
   0.269952     6 ->   8
   0.958312     6 ->  12
  Ground state amplitude: -0.000000

Percentage Active Character     99.78

  Amplitude    Excitation in Canonical Basis
   0.218969     0 ->   7
   0.110818     0 ->   8
  -0.210211     0 ->  11
   0.940678     0 ->  12

IROOT=  5: 20.091389 au   546.714 eV 4409550.1 cm**-1
  Amplitude    Excitation
   0.269914     6 ->   7
   0.958327     6 ->  11
  Ground state amplitude: -0.000000

Percentage Active Character     99.78

  Amplitude    Excitation in Canonical Basis
   0.110818     0 ->   7
  -0.218969     0 ->   8
  -0.940678     0 ->  11
  -0.210211     0 ->  12
```

MultiCore excitations are also available by setting 'CoreOrb[0]=0,1'. In following example, we directly access the oxygen 1S and carbon 1S core excitations

```
!STEOM-CCSD ExtremeSCF cc-pCVTZ Bohrs NoFrozencore

%mdci
nroots 5
FollowCIS true
DoSimpleDens False # use exact STEOM transition moment
maxiter 500
DoCVS true
DoCore true
CoreOrb[0]=0,1
end

*xyz 0 1
O 0 0 0.913973
C 0 0 -1.218243
*
```

It will give the oxygen and carbon K-edge spectrum as follows

```
-------------------------------
STEOM-CCSD RESULTS (SINGLETS)
-------------------------------

IROOT=  1: 10.577651 au   287.833 eV 2321526.1 cm**-1
  Amplitude    Excitation
  -0.508570     6 ->   7
   0.849534     6 ->   8
   0.127575     6 ->  10
  Ground state amplitude: -0.000000

Percentage Active Character     99.67

  Amplitude    Excitation in Canonical Basis
   0.803788     1 ->   7
   0.418299     1 ->   8
  -0.381488     1 ->  12
  -0.116973     1 ->  15

IROOT=  2: 10.577651 au   287.833 eV 2321526.1 cm**-1
  Amplitude    Excitation
   0.849534     6 ->   7
   0.508570     6 ->   8
   0.127575     6 ->  11
  Ground state amplitude:  0.000000

Percentage Active Character     99.67

  Amplitude    Excitation in Canonical Basis
   0.418299     1 ->   7
  -0.803788     1 ->   8
   0.381488     1 ->  11
   0.116973     1 ->  14

IROOT=  3: 10.900722 au   296.624 eV 2392431.9 cm**-1
  Amplitude    Excitation
  -0.893589     6 ->   9
   0.426665     6 ->  12
  Ground state amplitude:  0.000202

Percentage Active Character     79.85

Warning:: the state may have not converged with respect to active space
```

(continues on next page)

(continued from previous page)

```
-------------------- Handle with Care --------------------

  Amplitude    Excitation in Canonical Basis
  -0.972609     1 ->   9
   0.135434     1 ->  10
  -0.157896     1 ->  18

IROOT=  4: 19.690061 au   535.794 eV 4321468.9 cm**-1
  Amplitude    Excitation
   0.677274     5 ->   7
  -0.726090     5 ->   8
  Ground state amplitude: -0.000000

Percentage Active Character     98.89

  Amplitude    Excitation in Canonical Basis
  -0.733267     0 ->   7
  -0.606483     0 ->   8
   0.247689     0 ->  12
   0.137742     0 ->  15

IROOT=  5: 19.690061 au   535.794 eV 4321468.9 cm**-1
  Amplitude    Excitation
  -0.726090     5 ->   7
  -0.677274     5 ->   8
  Ground state amplitude:  0.000000

Percentage Active Character     98.89

  Amplitude    Excitation in Canonical Basis
  -0.606483     0 ->   7
   0.733267     0 ->   8
  -0.247689     0 ->  11
  -0.137742     0 ->  14
```

The core-valence separation should be used similar to that in the core-ionization. The only difference is that the natural orbital based active space selection scheme in STEOM-CCSD always rotate the particular core orbital to the HOMO. Now ORCA will automatically set the core orbitals to be the HOMO in STEOM-CCSD irrespective of the core-hole. One should use the exact STEOM-CCSD transition moment by using DoSimpleDens False. Fig. 5.32 presents the STEOM-CCSD oxygen K-edge spectra in thymine.

Fig. 5.31: Comparison of theoretical and experimental X-ray absorption spectra of oxygen K-edge in thymine. The simulated spectrum is shifted by -3.7 eV to align with the experimental spectrum.

One can interpret the results in terms of NTOs caculated from STEOM-CC eigen vectors

Fig. 5.32: Natural transition orbitals (ntos) for the oxygen K edge spectrum of thymine. All the core EE values mentioned are in eV and provided in the format (EE,Oscillator Strength).

5.15.4 Core-Excitation (UHF)

ORCA is now able to also perform core-excitations using the open-shell STEOM-CCSD method in a similar way to that of closed-shell.[671] By setting 'CoreOrb' for each spin channel we can perform a single core or a multicore calculation. In the following example, we determine the nitrogen and oxygen K-Shell excitations in the NO radical.

```
!UHF STEOM-CCSD ExtremeSCF nofrozencore usesym
!cc-pCVDZ

%mdci
nroots 10
DoCVS true
DoCore true
CoreOrb[0]=0,1        #N and O core orbitals (spin up)
CoreOrb[1]=0,1  #N and O core orbitals (spin down)
end

*xyz 0 2
  N          0.00000000000000      0.00000000000000      0.00490803701002
  O          0.00000000000000      0.00000000000000      1.14109196298998
*
```

and its corresponding output

```
----------------------
UHF STEOM-CCSD RESULTS
----------------------

IROOT=  1: 14.743862 au   401.201 eV 3235903.7 cm**-1 <S**2>= 0.877361  Sym: A2
```

(continues on next page)

(continued from previous page)

```
  Amplitude    Excitation
   0.637991     7a ->   8a
   0.750732     6b ->   7b
   0.136839     6b ->  10b

Percentage Active Character     99.93

  Amplitude    Excitation in Canonical Basis
  -0.613647     1a ->   8a
  -0.186064     1a ->  11a
  -0.722892     1b ->   7b
  -0.225768     1b ->  10b

IROOT=  2: 14.786433 au   402.359 eV 3245247.0 cm**-1 <S**2>= 0.849392  Sym: A1
  Amplitude    Excitation
   0.985414     6b ->   8b
   0.154505     6b ->  11b

Percentage Active Character     99.49

  Amplitude    Excitation in Canonical Basis
  -0.954188     1b ->   8b
   0.267334     1b ->  12b
  -0.121297     1b ->  16b

IROOT=  3: 14.794166 au   402.570 eV 3246944.1 cm**-1 <S**2>= 2.669487  Sym: A2
  Amplitude    Excitation
  -0.772273     7a ->   8a
   0.623458     6b ->   7b

Percentage Active Character     99.47

  Amplitude    Excitation in Canonical Basis
   0.754282     1a ->   8a
   0.166288     1a ->  11a
  -0.611046     1b ->   7b
  -0.132336     1b ->  10b

IROOT=  4: 15.243078 au   414.785 eV 3345469.0 cm**-1 <S**2>= 1.387083  Sym: B1
  Amplitude    Excitation
  -0.682643     7a ->   9a
  -0.101529     7a ->  11a
  -0.697873     6b ->   9b
  -0.133350     6b ->  12b

Percentage Active Character     98.11

  Amplitude    Excitation in Canonical Basis
   0.573243     1a ->   9a
   0.237647     1a ->  12a
  -0.265168     1a ->  13a
  -0.165128     1a ->  14a
   0.583678     1b ->   9b
   0.348965     1b ->  11b
   0.201568     1b ->  14b

IROOT=  5: 15.285144 au   415.930 eV 3354701.4 cm**-1 <S**2>= 2.434830  Sym: B1
  Amplitude    Excitation
  -0.690438     7a ->   9a
  -0.146046     7a ->  13a
   0.676597     6b ->   9b
```

(continues on next page)

(continued from previous page)

```
   0.173127     6b ->  13b

Percentage Active Character     93.48


------------------------- Handle with Care ----------------------------

Warning:: State   5 may have not converged with respect to active space
----------------------------------------------------------------------
  Amplitude    Excitation in Canonical Basis
   0.645706     1a ->   9a
   0.148300     1a ->  12a
  -0.211256     1a ->  13a
  -0.147915     1a ->  14a
  -0.643471     1b ->   9b
  -0.207585     1b ->  11b
  -0.158649     1b ->  14b

IROOT=  6: 19.682450 au   535.587 eV 4319798.5 cm**-1 <S**2>= 2.233840  Sym: A2
  Amplitude    Excitation
  -0.952369     6a ->   8a
  -0.128170     6a ->  10a
  -0.262026     5b ->   7b

Percentage Active Character     99.52

  Amplitude    Excitation in Canonical Basis
  -0.940546     0a ->   8a
   0.202713     0a ->  16a
  -0.254077     0b ->   7b

IROOT=  7: 19.714860 au   536.469 eV 4326911.7 cm**-1 <S**2>= 1.012274  Sym: A2
  Amplitude    Excitation
  -0.427285     6a ->   8a
   0.895703     5b ->   7b

Percentage Active Character     99.27

  Amplitude    Excitation in Canonical Basis
  -0.425513     0a ->   8a
   0.883394     0b ->   7b
  -0.176873     0b ->  15b

IROOT=  8: 19.746849 au   537.339 eV 4333932.5 cm**-1 <S**2>= 0.787890  Sym: A1
  Amplitude    Excitation
   0.992767     5b ->   8b

Percentage Active Character     99.45

  Amplitude    Excitation in Canonical Basis
   0.973851     0b ->   8b
   0.215849     0b ->  16b

IROOT=  9: 20.119392 au   547.476 eV 4415696.1 cm**-1 <S**2>= 2.441183  Sym: B1
  Amplitude    Excitation
   0.771305     6a ->   9a
  -0.127276     6a ->  11a
   0.122051     6a ->  13a
   0.592132     5b ->   9b

Percentage Active Character     96.84
```

5.16 Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program

Deprecated since ORCA 6.0.0

- The `orca_asa` program is no longer supported. It is still included in the 6.0.0 release and the documentation is preserved below. However, it may not function correctly and will be removed in a future ORCA version!
- The `!NMScan` and `!NMGrad` keywords are still available but these calculations may fail or not generate valid input for `orca_asa`. Please use the *ESD module* instead, if applicable.

In this section various aspects of the simulation and fit of optical spectra, including absorption, fluorescence, and resonance Raman are considered. This part of the ORCA is fairly autonomous and can also be used in a data analysis context, not only in a "quantum chemistry" mode. The program is called `orca_asa`, where ASA stands for "Advanced Spectral Analysis". The program was entirely designed by Dr. Taras Petrenko.

The general philosophy is as follows: An ORCA run produces the necessary data to be fed into the `orca_asa` program and writes an initial input file. This input file may be used to directly run `orca_asa` in order to predict an absorption, fluorescence or resonance Raman spectrum. Alternatively, the input file may be edited to change the parameters used in the simulations. Last – but certainly not least – the `orca_asa` program can be used to perform a fit of the model parameters relative to experimental data.

All examples below are taken from [672], which must be cited if you perform any work with the `orca_asa` program!

5.16.1 General Description of the Program

The program input comprises the following information: (1) model and specification of the model parameters characterizing the electronic structure of a molecule, as well as lineshape factors; (2) spectral ranges and resolution for simulations; (3) specification of vibrational transitions for rR excitation profile and spectra generation; (4) certain algorithm-selecting options depending on the model; (5) fitting options.

All optional parameters (1)-(3) are given in the `%sim` block, and fitting options are in the `%fit` block. The model parameters are specified within various blocks that will be described below. The program `orca_asa` is interfaced to ORCA and inherits its input style. The input for `orca_asa` run can be also generated upon ORCA run.

The current implementation features so called "simple", "independent mode, displaced harmonic oscillator" (IMDHO), and "independent mode, displaced harmonic oscillator with frequency alteration" (IMDHOFA) models.

5.16.2 Spectral Simulation Procedures: Input Structure and Model Parameters

Example: Simple Mode

This model represents the simplest approach which is conventionally used in analysis of absorption spectra. It neglects vibrational structure of electronic transitions and approximates each individual electronic band by a standard lineshape, typically a Gaussian, Lorentzian or mixed (Voigt) function. This model can only make sense if vibrational progressions are not resolved in electronic spectra. Upon this approximation the intensity of absorption spectrum depends on the energy of the incident photon (E_L), the electronic transition energy (E_T), the transition electric dipole moment (**M**, evaluated at the ground-state equilibrium geometry). Lineshape factors are specified by homogeneous linewidth Γ and standard deviation parameter σ corresponding to Gaussian distribution of transition energies. The following example illustrates a simple input for simulation of absorption bandshapes using various intensity and lineshape parameters.

```
# example001.inp
#
# Input file to generate absorption spectrum consisting
# of 3 bands   with different lineshape factors:
#
# 1. Lorentzian centered at 18000cm**-1 (damping factor Gamma= 100 cm**-1)
# 2. Gaussian   centered at 20000cm**-1
#    (standard deviation Sigma= 100 cm**-1)
# 3. Mixed Gaussian-Lorentzian band representing Voigt profile
#    centered at 21000 cm**-1

%sim
         Model Simple

         # Spectral range for absorption simulation:
         AbsRange     17000.0,   23000.0

         # Number of points to simulate absorption spectrum:
         NAbsPoints  2000

end


#-----------------------------------------------------------------------------
#  Transition        Gamma    Sigma  Transition Dipole Moment (atomic unit)
#  Energy (cm**-1)  (cm**-1) (cm**-1)      Mx          My        Mz
#-----------------------------------------------------------------------------

$el_states
3     # number of electronic states
   1  18000.0        100.00      0.0        1.0         0.0       0.0
```

(continues on next page)

(continued from previous page)

```
  2  20000.0          0.00    100.0        1.0         0.0       0.0
  3  22000.0         50.00     50.0        1.0         0.0       0.0
```

The parameters of of the final electronic states reached by the respective transitions are specified in the `$el_states` block. The spectral range and resolution used in the calculation are defined by the `AbsRange` and `NAbsPoints` keywords in `%sim` block. The calculation of the absorption spectrum is automatically invoked if `NAbsPoints>1`. After the `orca_asa` run you will find in your directory file `example001.abs.dat` containing absorption spectrum in simple two-column ASCII format suitable to be plotted with any spreadsheet program. Absorption spectra corresponding to individual electronic transitions are stored in file `example001.abs.as.dat` (the suffix "`as`" stands for "All States").

Fig. 5.33: Absorption spectrum generated after orca_asa run on file `example001.inp`. Three bands have different lineshape pararameters. Note that although all transitions are characterized by the same transition electric dipole moment their intensities are scaled proportionally to the transition energies.

The output of the program run also contains information about oscillator strengths and full-width-half-maximum (FWHM) parameters corresponding to each electronic band:

```
-----------------------------------------------
   State       EV        fosc    Stokes shift
            (cm**-1)               (cm**-1)
-----------------------------------------------
    1:      18000.00   0.054676      0.00
    2:      20000.00   0.060751      0.00
    3:      22000.00   0.066826      0.00
```

(continues on next page)

(continued from previous page)

```
---------------------------------------
   BROADENING PARAMETETRS (cm**-1)
---------------------------------------
   State     Gamma     Sigma      FWHM
---------------------------------------
    1:        100.00      0.00    200.00
    2:          0.00    100.00    235.48
    3:         50.00     50.00    180.07
```

Note that although all three types of lineshape functions are symmetric this is not true for the overall shapes of individual absorption bands since the extinction coefficient (absorption cross-section) is also proportional to the incident photon energy. Therefore, if the linewidth is larger than 10% of the peak energy the asymmetry of the electronic band can be quite noticeable.

Example: Modelling of Absorption and Fluorescence Spectra within the IMDHO Model

The IMDHO model is the simplest approach that successfully allows for the prediction of vibrational structure in electronic spectra as well as rR intensities for a large variety of real systems. This model assumes:

1. harmonic ground- and excited-state potential energy surfaces;
2. origin shift of the excited-state potential energy surface relative to the ground-state one;
3. no vibrational frequency alteration or normal mode rotation occurs in the excited state;
4. no coordinate dependence of the electronic transition dipole moment.

In addition to the parameters that enter the "Simple model" defined above it requires some information about the vibrational degrees of freedom. The required information consists of the ground-state vibrational frequencies $\{\omega_{gm}\}$ and (dimensionless) origin shifts $\{\Delta_{mi}\}$, where i and m refer to electronic states and normal modes respectively. Δ is expressed in terms of dimensionless normal coordinates. Accordingly, for the IMDHO model one has to specify the following blocks

- The `$el_states` block contains the parameters $E_T, \Gamma, \sigma, \mathbf{M}$ for each electronic state. By default E_T is assumed to be adiabatic minima separation energy. Alternatively, it can be redefined to denote for the vertical transition energy.This is achieved by specifiying the keyword `EnInput=EV` in the `%sim` block.
- A `$vib_freq_gs` block specifies ground-state vibrational frequencies.
- A `$sdnc` block contains parameters $\{\Delta_{mi}\}$ in matrix form such that the i-th column represents the dimensionless displacements along all normal modes for the i-th excited-state PES.

The file `example002.inp` provides the input for simulation of absorption and fluorescence spectra of a system characterized by significant displacements of the excited-state origin along 5 normal coordinates.

```
# example002.inp
#
# Input file for simulation  of vibrational structure
# in absorption and fluorescence spectra assuming
# origin shift of excited PES along 5 normal coordinates.
# The simulated spectra closely reproduce the experimental
# optical bandshapes for the tetracene molecule.
#
%sim
     Model IMDHO

     # spectral range for absorption simulation (cm**-1)
     AbsRange    20000.0, 27000.0
     NAbsPoints  2000             # number of points in absorption spectrum

     # spectral range for simulation of fluorescence (cm**-1)
     FlRange     22000.0, 16000.0
```

(continues on next page)

(continued from previous page)

```
     NFlPoints   2000      # number of points in fluorescence spectrum

     # the following options require the spectra to be normalized
     # so that their maxima are equal to 1.0
     AbsScaleMode Rel
     FlScaleMode  Rel   # default  for fluorescence

     # for absorption spectrum the default option is AbsScaleMode= Ext
     # which stands for extinction coefficient

end

#-----------------------------------------------------------------------------
#   Transition        Gamma    Sigma  Transition Dipole Moment (atomic unit)
#   Energy (cm**-1)  (cm**-1) (cm**-1)      Mx          My          Mz
#-----------------------------------------------------------------------------

$el_states
1
   1  21140.0         50.00    100.0        1.0         0.0         0.0


# Block specifying Stokes Shift parameter for each electronic state
# This information is optional
$ss
1   # number of excited states
  1   300.0      # the Stokes shift for the 1st electronic transition

# Block providing the values of VIBrational FREQuencies
# for 5 Ground-State normal modes.
# Obligatory for IMDHO and IMDHOFA models.
$vib_freq_gs
5
    1      310.0
    2     1193.0
    3     1386.0
    4     1500.0
    5     1530.0


# Block specifying origin Shift of the excite-state PES
# along each normal mode in terms of the ground-state
# Dimensionless Normal Coordinates
# Obligatory for IMDHO and IMDHOFA models.
$sdnc
5 1
                1
     1       0.698
     2      -0.574
     3       0.932
     4      -0.692
     5       0.561
```

The calculation of absorption and fluorescence spectra is automatically invoked if the parameters `NAbsPoints>1` and `NFlPoints> 1`. The input file also contains the optional block `$ss` which specifies the Stokes shift λ for each electronic transition. This parameter is equal to the energy separation between the 0-0 vibrational peaks in the absorption and fluorescence spectra as shown in Fig. 5.34 . In general λ accounts for solvent induced effects as well as unresolved vibrational structure corresponding to low-frequency modes that are not specified in the input. Note that we have specified parameters `AbsScaleMode=Rel` and `FlScaleMode=Rel` in `%sim` block in order to ensure that the simulated spectra are normalized to unity. The calculated absorption and fluorescence spectra are stored in `example002.abs.dat` and `example002.fl.dat` files, respectively.

Fig. 5.34: Absorption and fluorescence spectra generated after `orca_asa` run on the file `example002.inp`. If the homogeneous broadening is set to be $\Gamma = 10\,\mathrm{cm}^{-1}$ one can resolve underlying vibrational structure and identify various fundamental and combination transitions.

Example: Modelling of Absorption and Fluorescence Spectra within the IMDHOFA Model

IMDHOFA (Independent Mode Displaced Harmonic Oscillators with Frequency Alteration) is based on the same assumptions as the IMDHO model except for vibrational frequency alteration in excited state can take place. The file `example003.inp` features almost the same input parameters as `example002.inp`. The IMDHOFA model is invoked by the keyword `Model=IMDHOFA` in the `%sim` block. Additionally, one has to provide the obligatory block `$vib_freq_es`. It contains the excited-state vibrational frequencies $\{\omega_{emi}\}$ in matrix form such that the i-th column represents the vibrational frequencies of all normal modes for the i-th excited-state PES.

```
# Block providing the values of VIBrational FREQuencies
# for 5 Excited-State normal modes.
# Obligatory for IMDHOFA model.

$vib_freq_es
5  1          # number of modes and number of excited states
              1
    1       410.0
    2      1293.0
    3      1400.0
    4      1600.0
    5      1730.0
```

Fig. 5.35: Absorption and fluorescence spectra generated after `orca_asa` run on the file `example003.inp`. Also, the high-resolution spectra corresponding to homogeneous broadening $\Gamma = 10\,\mathrm{cm}^{-1}$ are shown.

Example: Modelling of Effective Broadening, Effective Stokes Shift and Temperature Effects in Absorption and Fluorescence Spectra within the IMDHO Model

For the IMDHO model the `orca_asa` is capable to model absorption and emission spectra in the finite-temperature approximation. While the keyword `Model=IMDHO` assumes the zero-temperature approximation, the value of `Model=IMDHOT` invokes the calculation of the spectra for the finite temperature which is specified by the paramter `TK` in the block `%sim`:

```
# example004.inp
#
#
%sim
     Model IMDHOT
     TK 300 # temperature (in Kelvin)

     # spectral range for absorption simulation (cm**-1)
     AbsRange    18000.0, 35000.0
     NAbsPoints  5000             # number of points in absorption spectrum

     # spectral range for simulation of fluorescence (cm**-1)
     FlRange     22000.0, 10000.0
     NFlPoints   5000      # number of points in fluorescence spectrum

     # the following options require the spectra to be normalized
```

(continues on next page)

(continued from previous page)

```
     # so that their maxima are equal to 1.0
     AbsScaleMode Rel
     FlScaleMode  Rel   # default  for fluorescence
end

#-----------------------------------------------------------------------------
#   Transition        Gamma    Sigma  Transition Dipole Moment (atomic unit)
#   Energy (cm**-1)  (cm**-1) (cm**-1)      Mx          My          Mz
#-----------------------------------------------------------------------------

$el_states
1
   1  21140.0         50.00    100.0        1.0         0.0         0.0


# Block specifying Stokes Shift parameter for each electronic state
$ss
1   # number of excited states
  1   300.0      # the Stokes shift for the 1st electronic transition

# Block providing the values of VIBrational FREQuencies
# for 10 Ground-State normal modes.
$vib_freq_gs
10
    1       30.0
    2       80.0
    3       100.0
    4       120.0
    5       130.0
    6      140.0
    7      160.0
    8      200.0
    9      310.0
   10     1300.0

# Block specifying origin Shift of the excite-state PES
# along each normal mode in terms of the ground-state
# Dimensionless Normal Coordinates
$sdnc
 10  1
              1
     1       2.5
     2       2.0
     3       1.8
     4       1.9
     5       1.5
     6       1.9
     7       2.4
     8       1.9
     9       2.5
    10       0.9
```

This example illustrates a typical situation in large molecules which feature a number of low frequency modes with significant values of dimensionless displacements for a given excited-state PES. In the case of high density of vibrational states with frequencies below or comparable to the intrincic value of FWHM (determined by Γ and σ) the vibrational progression is unresolved, whereby the spectra become very diffuse and show large separation between the maxima of absorption and emission spectra (Fig. 5.35). Besides, upon the condition $h\nu_i \leqslant kT$ the effective bandwidths and positions of maxima in the spectra can be strongly subject to temperature effects.

Fig. 5.36: Absorption and fluorescence spectra for T=0 K (blue) and T=300 K (red) generated after `orca_asa` run on the file `example004.inp`. Black lines show spectra corresponding to the case where all low-frequency modes were excluded from the calculation.

The effective Stokes shift and linewidth parameters which are evaluated in the simple self-consistent procedure are given in the output of the `orca_asa` run:

```
-------------------------------------------------------------------------------
State      E0           EV       fosc    Stokes shift  Effective Stokes shift
          (cm**-1)     (cm**-1)              (cm**-1)           (cm**-1)
-------------------------------------------------------------------------------
    1:     21140.00   24535.85   0.074529    300.00             7091.70


-------------------------------------------------------------------------------
↪------------
                                BROADENING PARAMETETRS (cm**-1)
-------------------------------------------------------------------------------
↪------------
                Intrinsic                                   Effective
 State   --------------------------  ---------------------------------------------
↪----------
                                                      Sigma                       ␣
↪FWHM
        Gamma    Sigma    FWHM      ---------------------------   -----------------
↪----------
                                          0K     298.15K   300.00K      0K     298.
↪15K   300.00K
-------------------------------------------------------------------------------
↪------------
  1:      50.00   100.00   293.50     1125.34   1411.13   1413.57   2703.84   3376.
↪75   3382.48
```

Note that the evaluation of the effective parameters is rather approximate and these values can noticeable deviate from those which can be directly deduced from the calculated spectra. However, such an information usually provides the proper order of magnitude of the effective vibronic broadening and Stokes shift. As indicated in the program output above, the effective bandshape has predominantly a Gaussian character which varies with the temperature so that $\sigma = 1125\,\mathrm{cm}^{-1}$ ($T = 0$ K) and $\sigma = 1414\,\mathrm{cm}^{-1}$ ($T = 300$ K). Indeed, as shown in Fig. 5.37 the absorption spectrum at $T = 300$ K can be well fitted using Gaussian lineshape with $\sigma = 1388\,\mathrm{cm}^{-1}$ (FWHM= $3270\,\mathrm{cm}^{-1}$). One can see that at higher temperatures the deviation between the spectrum and its Gauss fit becomes even smaller.

In molecules the normal distribution of the electronic transition energies in the ensemble would give rise to a Gaussian bandshape of the absorption band. However, the corresponding standard deviation is expected to be of the order of 100 cm^{-1}, whereby a typical Gaussian bandwidth of the order of 1000 cm^{-1} appears to result from unresolved vibronic progression. In general, this statement is supported by quantum chemical calculation of the model parameters. In principle the effective bandwidth parameters can also be used for characterization and assignement of individual electronic bands.

Fig. 5.37: Absorption spectrum (blue) for $T = 300$ K generated after `orca_asa` run on the file `example004.inp`. Red line represents the Gauss-fit of the calculated spectrum.

Example: Modelling of Absorption and Resonance Raman Spectra for the 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_u$ Transition in *trans*-1,3,5-Hexatriene

The hexatriene molecule is characterized by 9 totally-symmetric normal modes which dominate vibrational structure in absorption and are active in rR spectra corresponding to the strongly dipole-allowed $1-^1A_g \rightarrow 1-^1B_u$ transition around 40000 cm^{-1} . Except for some peculiarities related to the neglect of normal mode rotations in the excited state the optical spectra are quite satisfactorily described by the IMDHO model.

The following input exemplifies simulation of absorption spectrum and rR spectra for an arbitrary predefined number of excitation energies.

```
#
# example005.inp
```

(continues on next page)

(continued from previous page)

```
#
# input for simulation of absorption and resonance Raman spectra
# using experimental values of transition energy and displacement
# parameters corresponding to the strongly allowed 1-1Ag 1-1Bu transition
# in trans-1,3,5-hexatriene
#
%sim
   Model IMDHO

   AbsRange     38000.0,   48000.0
   NAbsPoints   2000
   AbsScaleMode Rel

   # resonance Raman intensities will be calculated
   # for all vibrational states with excitation number
   # up to RamanOrder:
   RamanOrder  4

   # excitation energies (cm**-1) for which rR spectra will be calculated:
   RRSE 39500, 39800, 41400

   # full width half maximum of Raman bands in rR spectra (cm**-1):
   RRS_FWHM  10

   RSRange   0, 5000  # spectral range for simulation of rR spectra (cm**-1)
   NRRSPoints  5000   # number of points to simulate rR spectra (cm**-1)

end


$el_states
1
   1  39800.0      150.00    0.0     1.0   0.0   0.0


$vib_freq_gs
9
    1      354.0
    2      444.0
    3      934.0
    4     1192.0
    5     1290.0
    6     1305.0
    7     1403.0
    8     1581.0
    9     1635.0

$sdnc
9 1
                1
      1       0.55
      2       0.23
      3       0.23
      4       0.82
      5       0.485
      6       0.00
      7       0.085
      8       0.38
      9       1.32
```

After the `orca_asa` run the following files will be created:

- `example005abs.dat` contains the simulated absorption spectrum. It is shown in Fig. 5.38.
- `example005.o4.rrs.39500.dat`, `example005.o4.rrs.39800.dat` and `example005.o4.rrs.41400.dat` contain the simulated rR spectra for excitation energies at 39500, 39800 and 41400 cm^{-1}, respectively. The suffix "`o4`" stands for the order of Raman scattering specified in the input by keyword `RamanOrder=4`. The rR specta are shown in Fig. 5.39.
- `example005.o4.rrs.39500.stk`, `example005.o4.rrs.39800.stk` and `example005.o4.rrs.41400.stk` provide Raman shifts and intensities for each vibrational transition. Corresponding vibrational states are specified by the quantum numbers of excited modes.

Fig. 5.38: Absorption spectrum corresponding to $1-^1 A_g \rightarrow 1-^1 B_u$ transition in *trans*-1,3,5-hexatriene generated after `orca_asa` run on the file `example005.inp`.

Fig. 5.39: Resonance Raman spectra for 3 different excitation energies which fall in resonance with $1-^1 A_g \rightarrow 1-^1 B_u$ transition in *trans*-1,3,5-hexatriene.

Note

- By default the program provides rR spectra on an arbitrary scale since only relative rR intensities within a

single rR spectrum are of major concern in most practical cases. However, one can put rR spectra corresponding to different excitation energies on the same intensity scale by providing the keyword `RSISM=ASR` in `%sim` block (RSISM – Raman Spectra Intensity Scaling Mode; ASR – All Spectra Relative). By default `RSISM=SSR` (SSR – Single Spectrum Relative) for which each rR spectrum is normalized so that the most intense band in it has intensity 1.0. The relative intensities of bands in rR spectra measured for different excitation energies can be compared if they are appropriately normalized relative to the intensity of a reference signal (e.g. Raman band of the solvent). We also keep in mind the possibility to extend our methodology in order to provide the absolute measure of rR intensities in terms of the full or differential cross-sections.

- Within the harmonic model, for a single electronic state neither relative rR intensities nor absorption bandshapes in the case of `AbsScaleMode=Rel` do depend on the values of the electronic transition dipole moment (unless it is precisely zero).

In the example above resonance Raman spectra have been generated for all vibrational transitions with total excitation number up to the value specified by the parameter `RamanOrder`. Its is also possible to make explicit specification of vibrational states corresponding to various fundamental, overtone and combination bands via the `$rr_vib_states` block. In such a case rR spectra involving only these vibrational transitions will be generated separately.

```
$rr_vib_states 5 # total number of vibrational transitions
 1
    modes 1
    quanta 1; # final vibrational state for the fundamental band corresponding to␣
↪mode 1
 2
    modes 9
    quanta 1; # final vibrational state for the fundamental band corresponding to␣
↪mode 9
 3
    modes 3, 4
    quanta 1, 1; # final vibrational state for the combination band involving␣
↪single
                 # excitations in modes 3 and 4
 4
    modes 5
    quanta 3; # final vibrational state for the second overtone band corresponding␣
↪to
              # mode 5
 5
     modes 1, 5,9
     quanta 1,2, 1; # final vibrational state for the combination band involving␣
↪single
                    # excitations in modes 1 and 2, and double excitation in mode 5
```

Each vibrational transition is specified via the subblock which has the following structure:

```
k
          modes m1,m2,...mn
          quanta q1,q2,...qn;
```

This means that the k-th transition is characteriezed by excitation numbers q_i for modes m_i so that corresponding Raman shift is equal to $\nu = \sum q_i \nu_i$, where ν_i is vibrational frequency of the mode m_i.

After the `orca_asa` run the following files will be created in addition:

- `example005.us.rrs.39500.dat`, `example005.us.rrs.39800.dat` and `example005.us.rrs.41400.dat` contain the simulated rR spectra involving only vibrational transitions specified in the `$rr_vib_states` block, for excitations energies at 39500, 39800 and 41400 cm^{-1}, respectively. The suffix "us" stands for "User specified vibrational States".
- `example005.us.rrs.39500.stk`, `example005.us.rrs.39800.stk` and `example005.`

us.rrs.41400.stk provide Raman shifts and intensities for each vibrational transition specified in the $rr_vib_states block.

Example: Modelling of Absorption Spectrum and Resonance Raman Profiles for the 1-^1A_g → 1-^1B_u Transition in *trans*-1,3,5-Hexatriene

The following example illustrates an input for simulation of absorption bandshape and resonance Raman profiles (RRP):

```
#
# example006.inp
#
# input for simulation of absorption and resonance Raman profiles
# using experimental values of transition energy and displacement
# parameters corresponding to the strongly allowed 1-1Ag 1-1Bu transition
# in trans-1,3,5-hexatriene
#

%sim
     Model IMDHO

     AbsRange     38000.0,   48000.0
     NAbsPoints   2000
     AbsScaleMode Rel

     RRPRange   38000.0, 48000.0 # spectral range for simulation of
                                 # rR profiles (cm**-1)

     NRRPPoints   2000 # number of points for simulation of rR profiles
     CAR 0.8

     RamanOrder 2
end

$el_states
1
   1  39800.0      150.00    0.0     1.0   0.0   0.0

$vib_freq_gs
9
    1      354.000000
    2      444.000000
    3      934.000000
    4     1192.000000
    5     1290.000000
    6     1305.000000
    7     1403.000000
    8     1581.000000
    9     1635.000000

$sdnc
9 1
                1
      1       0.55
      2       0.23
      3       0.23
      4       0.82
      5       0.485
      6       0.00
      7       0.085
      8       0.38
```

(continues on next page)

(continued from previous page)

```
      9       1.32

$rr_vib_states 5 # total number of vibrational transitions
 1
    modes 1
    quanta 1;
 2
    modes 9
    quanta 1;
 3
    modes 3, 4
    quanta 1, 1;
 4
    modes 5
    quanta 3;
5
     modes 1, 5,9
     quanta 1,2, 1;
```

The keyword `RamanOrder=2` will invoke generation of rR profiles for all vibrational transitions with total excitation number up to 2 in the range of excitation energies specified by the keywords `RRPRange` and `NRRPPoints`. Likewise, rR profiles for the vibrational states given in the `$rr_vib_states` block will be generated separately. Since in most cases only relative rR intensities are important, and one would be interested to compare absorption bandshape and shapes of individual rR profiles, the keyword `CAR = 0.8` is used to scale rR profiles for all vibrational transitions by a common factor in such a way that the ratio of the maximum of all rR intensities and the maximum of absorption band is equal to 0.8.

After the `orca_asa` run the following files will be created:

- `example006.abs.dat` contains the simulated absorption spectrum (Fig. 5.40).
- `example006.o1.rrp.dat` and `example006.o2.rrp.dat` contain rR profiles for vibrational transitions with total excitation numbers 1 and 2, respectively. RR profiles for all fundamental bands (from the file `example006.o1.rrp.dat`) are shown in Fig. 5.40.
- `example006.o1.info` and `example006.o1.info` contain specification of vibrational transitions with total excitation numbers 1 and 2, respectively, as well as corresponding Raman shifts.
- `example006.us.rrp.1.dat-example006.us.rrp.5.dat` contain rR profiles for vibrational transitions 1–5 specified in the `$rr_vib_states` block.

Fig. 5.40: Absorption spectrum and resonance Raman profiles of fundamental bands corresponding to $1 -^1 A_g \rightarrow 1 -^1 B_u$ transition in *trans*-1,3,5-hexatriene.

5.16.3 Fitting of Experimental Spectra

Example: Gauss-Fit of Absorption Spectrum

An absorption spectrum basically consists of a number of absorption bands. Each absorption band corresponds to a transition of the ground electronic state to an excited electronic state. In molecules such transitions are usually considerably broadened. In many cases there will be overlapping bands and one would need to deconvolute the broad absorption envelope into contributions from individual transitions. Within the "Simple model" the `orca_asa` program enables fit of an absorption spectrum with a sum of standard lineshape functions (Gaussian, Lorentzian) or more general Voigt functions. In most cases, one simply performs a "Gauss-Fit". That is, it is assumed that the shape of each individual band is that of a Gaussian function. Then one applies as many (or as few) Gaussians as are necessary for an accurate representation of the absorption envelope. In order to explain the fitting procedures within the "Simple model" let us consider an experimental absorption spectrum in Fig. 5.41:

Fig. 5.41: Experimental absorption spectrum. Bars indicate transition energies which were used for the initial guess in the input for spectral fitting.

As shown in Fig. 5.41 one can identify roughly 7 electronic bands. The initial estimates of transition energies corresponding to the maxima and shoulders in the absorption spectrum (indicated by bars in Fig. 5.41) and rather approximate values of inhomogeneous broadening and transition dipole moment components are specified in the `$el_states` block of the input file for the spectral fitting:

```
# example007.inp
#
# Input file for fitting of experimental absorption spectrum
#

%sim
        model Simple
end

%fit
       Fit true                # Global flag to turn on the fit
       AbsFit true             # Flag to include absorption into the fit
       method Simplex
       WeightsAdjust true

       AbsRange 0.0, 100000.0 # absorption spectral range to be included in the␣
↪fit;
                               # in the present case all experimental points
                               # will be included


       AbsName "absexp.dat"    # name of the file containing experimental
                               # absorption spectrum in a simple two-column
                               # ASCII format
```

(continues on next page)

(continued from previous page)

```
       ExpAbsScaleMode Ext    # This keyword indicates that the experimental
                              # absorption intensity is given in terms of
                              # the extinction coefficient. This is important
                              # for the proper fitting of transition dipole
                              # moments and oscillator strengths

       NMaxFunc 10000   # maximum number of function evaluations in simplex
                        # algorithm

       MWADRelTol 1e-5  # Relative Tolerance of the Mean Weighted Absolute
                        # Difference  (MWAD) function which specifies the
                        # convergence criterion


       E0Step 500.00    # initial step for the transition energies
                        # in the simplex fitting

       TMStep 0.5       # initial step for the transition dipole moments
                        # in the simplex fitting

       E0SDStep 500.0   # initial step for the inhomogeneous linewidth (Sigma)
                        # in the simplex fitting


end

# ! Parameters specified in the $el_states block
#   are used as initial guess in the fit

#-----------------------------------------------------------------------------
#  Transition        Gamma    Sigma  Transition Dipole Moment (atomic unit)
#  Energy (cm**-1)  (cm**-1) (cm**-1)      Mx          My        Mz
#-----------------------------------------------------------------------------

$el_states
7
    1 11270           0.0     1000.00    1.0000     0.0000     0.0000
    2 15100           0.0     1000.00    1.0000     0.0000     0.0000
    3 20230           0.0     1000.00    1.0000     0.0000     0.0000
    4 27500           0.0     1000.00    1.0000     0.0000     0.0000
    5 31550           0.0     1000.00    1.0000     0.0000     0.0000
    6 37070           0.0     1000.00    1.0000     0.0000     0.0000
    7 39800           0.0     1000.00    1.0000     0.0000     0.0000


# the integer values specified in $el_states_c block indicate parameters
# in the $el_states block to be varied
$el_states_c
7
    1    1             0          1          1          0          0
    2    2             0          2          2          0          0
    3    3             0          3          3          0          0
    4    4             0          4          4          0          0
    5    5             0          5          5          0          0
    6    6             0          6          6          0          0
    7    7             0          7          7          0          0
```

The functionality of the constraint block `$el_states_c` should be understood as follows: 1) 0 flag indicates that the corresponding parameter in the `$el_state` block will not be varied in the fitting; 2) if the number corresponding to a certain parameter coincides with the number of the corresponding electronic state this parameter will be varied independently. Thus, the block `$el_states_c` in the input indicates that all transition energies, inhomo-

geneous linewidths and x-components of the transition electric dipole moment will be varied independently, while homogeneous linewidths, y- and z-components of the transition dipole moment will be fixed to their initial values.

The following considerations are important:

- Since in conventional absorption spectroscopy one deals with the orientationally averaged absorption cross-section, the signal intensity is proportional to the square of the transition electric dipole moment $|\mathbf{M}|^2$. Thus, the intensities do not depend on the values of the individual components of $\mathbf{M}$ as long as $|\mathbf{M}|^2 = \text{const}$. Therefore, we have allowed to vary only M_x components. Otherwise there can be problems in convergence of the fitting algorithm.
- The sum of the weights of experimental points which enter the mean absolute difference function employed in the minimization is always kept equal to the number of experimental points. In the case of equidistant experimental photon energies all weights are assumed to be equal. However, in experimental electronic spectra the density of spectral points can increase significantly upon going from high- to low-energy spectral regions, which is due to the fact that experimental absorption spectra are initially acquired on the wavelength scale. In such a case the quality of the fit can be noticeably biased towards low-energy spectral region. Therefore, it is advisable to adjust relative weights of experimental points according to the their density which is controlled by the keyword `WeightsAdjust` in the `%fit` block. Although this parameter is not crucial for the present example, in general, it will provide a more balanced fit.
- The parameters `E0Step`, `TMStep`, `E0SDStep` in the `%fit` block specify the initial dimension of the simplex in the space of $E_T, \mathbf{M}, \sigma$ and should roughly correspond to the expected uncertainty of initial guess on these parameters in the `$el_states` block relative to their actual values. The quality of the fit can noticeably deteriorate if the parameters specifying initial steps are too low or too high.

The fit run of `orca_asa` on file `example007.inp` will converge upon approximately 3600 function evaluations (for `MWADRelTol=1e-5`). The results of the fit will be stored in file `example007.001.inp` which has the same structure as the input file `example007.inp`. Thus, if the fit is not satisfactory and/or it is not fully converged it can be refined in a subsequent `orca_asa` run upon which file `example007.002.inp` will be created, and so on. Some model parameters in intermediate files can be be additionally modified and/or some constraints can be lifted or imposed if so desired. The output file `example007.001.inp` will contain fitted model parameters stored in the `$el_states` block:

```
$el_states
7
   1 11368.24     0.00  732.50     1.6290   0.0000   0.0000
   2 15262.33     0.00  495.17    -0.2815   0.0000   0.0000
   3 19500.08     0.00  1023.39     0.2300   0.0000   0.0000
   4 26969.01     0.00  1832.30     1.4089   0.0000   0.0000
   5 31580.41     0.00  1440.87     1.8610   0.0000   0.0000
   6 35769.07     0.00  1804.02     1.5525   0.0000   0.0000
   7 39975.11     0.00  1909.38     2.4745   0.0000   0.0000
```

The overall quality of the fit is determined by the parameter MWAD which upon convergence reaches the value of ≈0.009 (MWAD stands for Mean Weighted Absolute Difference).

After the `orca_asa` run files `absexp.fit.dat` and `absexp.fit.as.dat` will be created. Both files contain the experimental and fitted spectra which are shown in Fig. 5.42 . In addition, the file `absexp.fit.as.dat` will contain individual contributions to the absorption spectrum corresponding to different excited states.

Fig. 5.42: Comparison of the experimental (black curve) and fitted (red) absorption spectra corresponding to the fit run of `orca_asa` on the file `example007.inp`. Blue curves represent individual contributions to the absorption spectrum from each state.

Since there is a noticeable discrepancy between the fitted and experimental spectra around 13000 cm^{-1} (Fig. 5.42) it is worthwhile to refine the fit after adding parameters for a new state in the file `example007.001.inp`:

```
$el_states
8
   1 11368.24     0.00  732.50     1.6290   0.0000   0.0000
... ... ...
   8 13280.00     0.00  1000.00     1.000   0.0000   0.0000


$el_states_c
8
   1     1          0         1          1         0         0
... ... ...
   8     8          0         8          8         0         0
```

Actually, the character of the discrepancy in the present case is very similar to that in Fig. 5.39 (section *Example: Modelling of Effective Broadening, Effective Stokes Shift and Temperature Effects in Absorption and Fluorescence Spectra within the IMDHO Model*) where a vibronically broadened absorption spectrum was fitted with a Gaussian lineshape. Thus, the poor fit in the region around 1300 cm^{-1} is most likely due to the essentially asymmetric character of the vibronic broadening rather than to the presence of another electronic band.

As shown in Fig. 5.43 the refined fit leads to much better agreement between the experimental and fitted absorption spectra (MWAD=0.0045).

Fig. 5.43: Comparison of the experimental (black) and fitted (red) absorption spectra corresponding to the fit run of `orca_asa` on the file `example007.001.inp`. Blue curves represent individual contributions to the absorption spectrum from each state.

Due to some peculiarities of the simplex algorithm for function minimization, you can still refine the fit by rerunning `orca_asa` on the file `example007.002.inp`! This leads to an even lower value of the parameter MWAD= 0.0038, and therefore to better agreement of experimental and fitted spectra (even though the previous run has been claimed to be converged).

It is also possible to perform a fit using the same value of inhomogeneous linewidth for all electronic states. For this purpose one needs to choose as a guess the same linewidth parameters in the `$el_states` block:

```
$el_states
8
   1 11118.58     0.00  1000.0      1.0687   0.0000   0.0000
   2 13673.38     0.00  1000.0     -0.5530   0.0000   0.0000
   3 21267.40     0.00  1000.0      0.3675   0.0000   0.0000
   4 27024.71     0.00  1000.0      1.4041   0.0000   0.0000
   5 31414.74     0.00  1000.0      1.7279   0.0000   0.0000
   6 35180.77     0.00  1000.0      1.6246   0.0000   0.0000
   7 39985.52     0.00  1000.0      2.5708   0.0000   0.0000
   8 11665.01     0.00  1000.0      1.2332   0.0000   0.0000
```

In addition the constraint block should be modified as follows:

```
$el_states_c
8
   1      1          0         1          1         0         0
   2      2          0         1          2         0         0
   3      3          0         1          3         0         0
   4      4          0         1          4         0         0
   5      5          0         1          5         0         0
```

(continues on next page)

(continued from previous page)

```
    6       6           0           1           6           0           0
    7       7           0           1           7           0           0
    8       8           0           1           8           0           0
```

The constraint parameters for the inhomogeneous broadening were chosen to be 1, which means that formally σ_1 corresponding to the first state is varied independently while the linewidths $\{\sigma_i\}$ for other bands are varied in such a way that the ratios σ_i/σ_1 are kept fixed to their initial values, whereby the same linewidth parameter will be used for all states.

Fig. 5.44: Comparison of the experimental (black) and fitted (red) absorption spectra corresponding to the fit run of `orca_asa` on the file `example007.002.inp` in which equal broadening was assumed for all electronic bands. Blue curves represent individual contributions to the absorption spectrum from each state.

One can see (Fig. 5.44) that the assumption of equal linewidths for all electronic bands leads to a rather pronounced deterioration of the quality of the fit in the low-energy spectral range (MWAD=0.017). Apparently, this discrepancy can be fixed assuming more electronic states at higher energies.

NOTE

- The homogeneous linewidth parameters can also be included in the fit in a similar way. However, one can see that in most cases they appear to be much smaller than corresponding Gaussian linewidth parameters.
- Gauss-fit of absorption spectra is coventionally performed assuming the same linewidth parameters for all bands. However, since a large portion of Gaussian broadening is mainly due to the unresolved vibronic structure in the spectra which can significantly vary depending on the nature of transition, the assumption of unequal Gaussian bandwidths seems to be a physical one.

Example: Fit of Absorption and Resonance Raman Spectra for 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_u$ Transition in *trans*-1,3,5-Hexatriene

Below we provide an example of the fit of the lineshape parameters and $\{\Delta_m\}$ corresponding to the strongly dipole-allowed 1-^{1}A$_g$ $\rightarrow$1-^{1}B$_u$ transition in hexatriene. It is known that the most intense bands in rR spectra correspond to the most vibronically active in absorption spectrum. For the IMDHO model this correlation is determined by the values of $\{\Delta_m\}$. Thus, the larger Δ, the larger is the rR intensity of a given mode and the more pronounced is the progression in the absorption spectrum corresponding to this mode. In principle, if all vibrational transitions in absorption are well resolved it is possible to determine $\{\Delta_m\}$ by a fit of the absorption spectrum alone. In practice this task is ambiguous due to the limited resolution of the experimental absorption spectra. The observation of a rR spectrum enables the identification of the vibrational modes that are responsible for the progression in the absorption spectrum, as well as a quantitative analysis in terms of $\{\Delta_m\}$. The file `example006.inp` provides a brute-force example on how to approach the fit employing the minimal possible experimental information: 1) An absorption spectrum; 2) relative rR intensities of fundamental bands for a given excitation energy. The rR spectrum upon the excitation in resonance with the 0-0 vibronic band at 39809 cm^{-1} is shown in Fig. 5.33.

Fig. 5.45: Experimental Resonance Raman spectrum corresponding to 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_u$ transition in *trans*-1,3,5-hexatriene.

The experimental rR spectrum has enabled the identification of seven vibrational modes that give rise to the most intense resonance Raman bands. Therefore, they are expected to have the largest excited-state displacements and the most pronounced effect on the vibrational structure of the absorption spectrum. Their vibrational frequencies have been entered as input for the fit as shown below:

```
#
# example008.inp
#
# Input for fit of absorption and resonance Raman spectra
# corresponding to the strongly allowed 1-1Ag 1-1Bu transition
# in 1,3,5 trans-hexatriene.
```

(continues on next page)

(continued from previous page)

```
#
# Parameters to be varied:
# 1) adiabatic minima transiton energy
# 2) homogeneous linewidth (Gamma)
# 3) dimensionless normal coordinate displacements of the
#    excited-state origin
#

%sim
      Model IMDHO
end


%fit

    Fit true     # boolean parameter to switch on the fit

    # boolean parameter to include experimental absorption
    # spectrum in the fit:
    AbsFit true

    # boolean parameter to include experimental rR spectra
    # specified in $rrs_exp block in the fit:
    RRSFit true

    AbsExpName "hex-abs.dat" # name of the file with experimental absorption
                             # spectrum

    # the following value of keyword ExpAbsScaleMode
    # indicates that only the shape of absorption band
    # but not its total intensity will be accounted in the fit:
    ExpAbsScaleMode Rel

    # the weight of absorption relative to the total weight of
    #  rR intensities in the difference function to be minimized:
    CWAR  5.0

    NMaxFunc  1000   # maximum number of function evaluations in simplex
                     # algorithm

    MWADRelTol 1e-4  # Relative Tolerance of the Mean Weighted Absolute
                     # Difference  (MWAD) function which specifies the
                     # convergence criterion
    SDNCStep  1.0

end


# The values specified in $el_states block serve as initial guess in the fit
$el_states
1
   1  40000.0      200.00    0.0     1.0   0.0   0.0

# the integer values specified in $el_states_c block indicate parameters
# in $el_states block to be varied
$el_states_c
1
   1     1            1       0       0     0     0

# 7 totally symmetric vibrations which give rise to the most
# intense bands in the rR spectra are included into analysis.
```

(continues on next page)

(continued from previous page)

```
# Experimental values of vibration frequencies are given:
$vib_freq_gs
7
    1      354.0
    2      444.0
    3      934.0
    4     1192.0
    5     1290.0
    6     1403.0
    7     1635.0

# Initial guess for the values of dimensionless normal
# coordinate displacements of the excited-state origin
$sdnc
7 1
                1
      1       0.0
      2       0.0
      3       0.0
      4       0.0
      5       0.0
      6       0.0
      7       0.0

# the integer values specified in $sdnc_c block indicate parameters
# in $sdnc block to be varied
$sdnc_c
7 1
              1
      1        1
      2        2
      3        3
      4        4
      5        5
      6        6
      7        7


# specification  of vibrational transitions and their intensities
# in experimental rR spectra:
$rrs_exp
1         # number of rR spectra
1 1       # start of the block specifying the 1st rR spectrum
    Ex 39809.0 # excitation energy for the first rR spectrum
    NTr  7     # number of vibrational transitions for which intensities are
               # provided
    1
       int   10.0  1.0
       modes  1
       quanta 1;
    2
       int   5.0  1.0
       modes  2
       quanta 1;
    3
       int   1.5  1.0
       modes  3
       quanta 1;
    4
       int   21.0  1.0
       modes  4
```

(continues on next page)

(continued from previous page)

```
      quanta 1;
   5
      int   7.5  1.0
      modes  5
      quanta 1;
   6
      int   2.0  1.0
      modes  6
      quanta 1;
   7
      int   46.0  1.0
      modes  7
      quanta 1;
```

The input of rR intensities for an arbitrary number of excitation energies follows the keyword `$rrs_exp` block:

```
$rrs_exp
1         # number of rR spectra
1 1
```

The first "1" in the last line denotes the number of the rR spectrum for which specification starts below. If the second number is the same as the number of the spectrum, then it means that only relative intensities for the first rR spectrum are meaningful in the fit. If several spectra are given in the input then the second number may have a different value, e.g.:

```
$rrs_exp
3         # number of rR spectra
1 2
...
```

This input is to be interpreted as indicating that 3 rR spectra are provided and the relative intensities for the first spectrum are given on the same scale as the second one that will be accounted for in the fit. The value of the excitation energies and the number of vibrational transitions specified are indispensable within the blocks specifying intensities for each rR spectrum.

Following the number of vibrational transitions given by the keyword `NTr` one has to specify each vibrational transition and its intensity. Thus, in the present case there are seven subblocks with the following structure:

```
k    int   I  W
          modes m1,m2,...mn
          quanta q1,q2,...qn;
```

This means that the k-th transition has intensity I and weight W in the mean absolute difference function that is used for the minimization (W is an optional parameter). The following 2 lines specify the vibrational transitions by providing excitation numbers q_i for modes m_i so that the corresponding Raman shift is equal to $\nu = \sum_{q_i \nu_i}$, where ν_i is vibrational frequency of the mode m_i.

The parameters that are to be varied are specified within the constraint blocks `$el_states_c` and `$sdnc_c`. Both blocks have the same structure and number of parameters as `$el_states` and `$sdnc`, respectively. A parameter from the `$el_states` block is supposed to be independently varied if its counterpart from the `$el_states_c` block is equal to the number of the electronic state. Likewise, a parameter from the `$sdnc` block is supposed to be independently varied if its counterpart from the `$sdnc_c` block is equal to the number of the normal mode. Model parameters that are set to 0 in the corresponding constraint blocks are not varied in the fit. The values of the following parameters may be important for the quality of the fit:

- `CWAR` in the `%fit` block specifies the weight of absorption relative to the weight of rR intensities in the difference function to be minimized. If this parameter was not specified the fit would be almost insensitive to the rR intensities in the input, since typically the number of experimental absorption points is much larger than the number of rR transitions in the input. In most cases the value of `CWAR` in the range 1.0–5.0 is a good choice since the error in the measured experimental intensity is expected to be much smaller for absorption

than for resonance Raman.

- SDNCStep in the %fit block specifies the initial dimension of the simplex in the space of $\{\Delta_m\}$ and should roughly correspond to the expected uncertainty of initial guess on $\{\Delta_m\}$ in the $sdnc block compared to their actual values. You can notice in the present example that if this parameter is too large (>2.0) or too small (<0.4) the quality of the fit may significantly deteriorate
- Although the default initial dimensions of the simplex have reasonable values for different types of parameters it may turn out to be helpful in some cases to modify the default values:

```
FREQGStep 10.0  # ground-state vibrational frequencies
         FREQEStep 10.0  # excited-state vibrational frequencies
         E0Step    300.0 # transition energies
         SSStep     20.0 # Stokes shift
         TMStep      0.5 # electronic transition dipole moment
         GammaStep  50.0 # homogeneous linewidth
         E0SDStep   50.0 # inhomogeneous linewidth
         SDNCStep 1.0 # origin shift along dimensionless normal coordinate
```

The fit run of orca_asa on the file example008.inp will converge upon approximately 700 function evaluations (for MWADRelTol=1e-4). The results of the fit will be stored in file example008.001.inp which has the same structure as the input file example008.inp. Thus, if the fit is not satisfactory and/or it is not fully converged it can be refined in subsequent orca_asa run upon which file example008.002.inp will be created, and so on. Some model parameters in intermediate files can be be additionally modified and/or some constraints can be lifted if so desired. The output file example008.001.inp will contain fitted displacement parameters $\{\Delta_m\}$stored in the $sdnc block:

```
$sdnc
7 1
                    1
      1       0.675000
      2      -0.194484
      3      -0.217527
      4       0.811573
      5       0.529420
      6      -0.149991
      7       1.314915
```

In the present example, these parameters are actually in very close agreement with those published for the hexatriene molecule!

The overall quality of the fit is determined by the parameter MWAD which upon convergence reaches the value of ≈0.027. The fitted rR intensities are presented in the commented lines next to the experimental rR intensities in file example008.001.inp:

```
$rrs_exp
1
   1   1 3.495285e+001
   Ex    39809.00
   NT   7
      1
        Int 10.0 1.0  # simulated intensity: 1.000982e+001
        modes  1
        quanta 1;
      2
        Int 5.0 1.0   # simulated intensity: 8.976285e-001
        modes  2
        quanta 1;
      3
        Int 1.5 1.0   # simulated intensity: 1.255880e+000
        modes  3
        quanta 1;
```

(continues on next page)

(continued from previous page)

```
    4
      Int 21.0 1.0  # simulated intensity: 1.761809e+001
      modes  4
      quanta 1;
    5
      Int 7.5 1.0   # simulated intensity: 7.499749e+000
      modes  5
      quanta 1;
    6
      Int 2.0 1.0   # simulated intensity: 6.014466e-001
      modes  6
      quanta 1;
    7
      Int 46.0 1.0  # simulated intensity: 4.600071e+001
      modes  7
      quanta 1;
```

The file hex-abs.fit.dat will contain the experimental and fitted absorption spectra in ASCII format which can be plotted in order to visualize the quality of absorption fit (Fig. 5.46).

Fig. 5.46: Experimental (black) and fitted (red) absorption spectrum corresponding to 1-^{1}A$_g$ → 1-^{1}B$_u$ transition in 1,3,5 *trans*-hexatriene.

Note

- The more experimental rR intensities are included in the analysis the more reliable is the fit. In principle it is possible to obtain fully consistent results even if only a limited number of vibrational transitions is provided. However, in such a case it is desirable to include into analysis at least a single Raman transition

involving the mode for which Δ is to be determined.

- The quality of the fit can be improved if the IMDHOFA model is invoked and excited-state vibrational frequencies are allowed to vary.
- Due to the initial guess and dimension of the simplex, as well as some peculiarities of the simplex algorithm for function minimization, you can still refine the fit by rerunning `orca_asa` on file `example008.001.inp` that may lead to an even lower value of the parameter `MWAD = 0.021`, and therefore to better agreement of experimental and fitted spectra (even though the previous run has been claimed to be converged).
- In this respect it appears to be wise to perform the fit in 3 steps:
  1. Fit the preresonance region below the 0-0 vibronic band with a single Lorentzian band, from which the adiabatic transition energy E_0, and homogeneous linewidth Γ are obtained. The range for fit of the absorption spectrum can be specified by the `AbsRange` keyword in the `%fit` block.
  2. Fix E_0 and Γ, and optimize $\{\Delta_m\}$ fitting the entire spectral range and rR intensities.
  3. Lift constraints on E_0 and Γ, and reoptimize simultaneously all parameters.

Example: Single-Mode Fit of Absorption and Fluorescence Spectra for 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_{2u}$ Transition in Tetracene

In this section we provide an example and discuss the most important aspects of joint fit of fluorescence and absorption spectra. Fig. 5.47 displays the experimental emission and absorption spectra corresponding to 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_{2u}$ transition in tetracene.

Fig. 5.47: Deconvoluted absorption (red) and fluorescence (blue) spectra of tetracene in cyclohexane upon the assumption of a single vibronically active mode. The black solid lines represent experimental spectra.

Both spectra show pronounced effective vibrational progressions that are dominated by 3 and 5 peaks, respectively.

As can be shown on the basis of quantum chemical calculations this progression has essentially multimode character. However, the experimental spectra can be well fitted under the assumption of a single vibronically active mode. The input has the following structure:

```
#
# example009.inp
#
# Parameters to be varied:
# 1) adiabatic minima transition energy
# 2) homogeneous and inhomogeneous linewidths
# 3) normal mode frequency and corresponding dimensionless displacement of the
#    excited-state origin
#

%sim
      Model IMDHO
      EnInput E0 # we assume adiabatic minima separation energies
end

%fit
    Fit true       # global flag to turn on the fit
    AbsFit true    # flag to include absorption spectrum into the fit
    FlFit true     # flag to include fluorescence spectrum into the fit

    WeightsAdjust true

    AbsRange 19000.0, 28000.0 # spectral range for absorption
                              # which will be included into the fit

    FlRange  17800.0, 22300.0 # spectral range for absorption
                              # which will be considered in the fit

    AbsName "absexp.dat" # name of the file containing experimental
                         # absorption spectrum in a simple two-column
                         # ASCII format
    FlName  "flexp.dat"  # name of the file with experimental fluorescence spectrum

    ExpAbsScaleMode Rel  # flags indicating that only relative shapes of the
    ExpFlScaleMode Rel   # absorption and fluorescence bands will be fitted.


    CWAF   1.000 # important parameter to have a balanced relative quality of fit
                 # of fluorescence and absorption

    NMaxFunc 10000  # maximum number of function evaluations in simplex
                    # algorithm

    MWADRelTol= 0.0001  # Relative Tolerance of the Mean Weighted Absolute
                        # Difference  (MWAD) function which specifies the
                        # convergence criterion

    TMStep 0.5       # initial step for the transition dipole moments
                     # in the simplex fitting

    E0SDStep 500.0   # initial step for the inhomogeneous linewidth (Sigma)

    FREQGStep 100.00 # initial step for the vibrational frequencies

    E0Step 1000.0    # initial step for the transition energies

    SSStep 10.0      # initial step for the Stokes shift
```

(continues on next page)

(continued from previous page)

```
    GammaStep 100    # initial step for the homogeneous linewidth

    SDNCStep 0.5    # initial step for the displacement parameter

end

$el_states
2
   1 21100.00    100.00   100.00     1.0000   0.0000   0.0000
   2 24000.00    100.00   1000.00    1.0000   0.0000   0.0000

$el_states_c
2
   1     1         1        1          0        0        0
   2     2         2        2          2        0        0

$abs_bool
2
  1  1
  2  1

$fl_bool
2
  1  1
  2  0

$ss
2
    1      100.000000
    2        0.000000

$ss_c
2
    1            1
    2            0


$vib_freq_gs
1
    1      1500.0


$vib_freq_gs_c
1
    1            1

$sdnc
1 2
                    1          2
      1       2.0000000   0.000000

$sdnc_c
1 2
                    1          2
      1         1          0
```

The parameter `CWAF=1.0` in the `%fit` block specifies the weight of absorption relative to the weight of fluorescence in the difference function to be minimized. If this parameter was not specified the quality of the fit would be biased towards the spectrum with a larger number of experimental points. In some typical situations where the error in the measured experimental intensity is expected to be smaller for absorption than for emission it is desirable to choose the value of `CWAF` to be more than 1.0.

In order to account for a broad featureless background signal in the absorption spectrum above 24000 cm^{-1}, the second band was included into the analysis and approximated with a Voigt lineshape which means also that the corresponding frequency in the `$vib_freq_gs` block and displacement parameter in the `$sdnc` block are fixed to zero in the fit. Thus, the `$el_states` block contains an initial guess on the transition energies, transition electric dipole moments and linewidth parameters for 2 states:

```
$el_states
2
   1 21100.00    100.00   100.00     1.0000   0.0000   0.0000
   2 24000.00    100.00   1000.00    1.0000   0.0000   0.0000
```

The initial value of the adiabatic minima separation energy for the first state was approximated by the energy corresponding to the first vibronic peak in the absorption spectrum (21100 cm^{-1}). The transition energies and linewidth parameters are varied independently as indicated in the `$el_states_c` block. Since we allow to fit only bandshapes, but not the overall intensities of the spectra, only relative absolute values of the transition electric dipole moments of two bands are important. Therefore it is reasonable to fix all components of the transition moment for the first state and vary only M_x component for the second one:

```
$el_states_c
2
   1     1         1        1          0        0        0
   2     2         2        2          2        0        0
```

Since we assume the absorption by both states and emission only from the first one, it is necessary to include Boolean arrays $abs_bool and $fl_bool which specify states which will be included in the treatment of the absorption and fluorescence spectra, respectively:

```
$abs_bool
2
  1  1 # 1 indicates that the corresponding state will be included in the␣
↪calculation of
  2  1 # absorption


$fl_bool
2
  1  1
  2  0 # 0 indicates that the corresponding state will be excluded from the␣
↪calculation
       # of emission spectrum
```

We need also to vary the value of vibrational frequency of the mode which determines separation of vibrational peaks in the spectra. This is done via the constraint block `$vib_freq_gs_c`:

```
$vib_freq_gs_c
1
    1            1
```

Note that it is meaningless to include into the treatment the Stokes shift for the second state which give rise to the background signal in the absorption since the corresponding emission is not present. Therefore λ for the second state is fixed to zero as indicated in the `$ss` block and its constraint counterpart `$ss_c`:

```
$ss
2
    1      100.000000 # initialization of the Stokes shift for the 1st electronic␣
↪state
    2        0.000000

$ss_c
2
    1         1 # the Stokes shift for the 1st electronic state will be varied in␣
```

(continues on next page)

(continued from previous page)

```
↪the fit
   2         0 # the Stokes shift for the 2nd electronic state will be fixed in␣
↪the fit
```

The fit run of `orca_asa` on file `example009.inp` will converge upon approximately 700 function evaluations (for `MWADRelTol=1e-4`). The file `example009.001.inp` will contain the fitted effective values of the vibrational frequency and dimensionless displacement: $\omega = 1404\,\mathrm{cm}^{-1}$, $\Delta = 1.35$. One can notice that the fit is rather poor in the low- and high-energy edges of the absorption and fluorescence spectra, respectively (Fig. 5.47). The source of this discrepancy is the single-mode approximation which was employed here. The quality of the fit can be significantly improved assuming several modes with non-zero displacement parameters. Note that in such a case the proper guess on the number of active modes and corresponding dimensionless displacements can be deduced from quantum chemical calculations.

5.16.4 Quantum-Chemically Assisted Simulations and Fits of Optical Bandshapes and Resonance Raman Intensities

In this section we finally connect the spectra simulation algorithms to actual quantum chemical calculations and outline a detailed approach for the analysis of absorption, fluorescence and resonance Raman spectra within the IMDHO model. Our procedure becomes highly efficient and nearly automatic if analytical excited state derivatives with respect to nuclear displacements are available. However, this availability is not mandatory and hence, spectral predictions may as well be achieved by means of normal mode scan calculations for high-level electronic structure methods for which analytic gradients have not been implemented.

Example: Quantum-Chemically Assisted Analysis and Fit of the Absorption and Resonance Raman Spectra for 1-^{1}A$_g$ $\rightarrow$ 1-^{1}B$_u$ Transition in *trans*-1,3,5-Hexatriene

The following input file for an ORCA run invokes the calculation of the excited-state origin displacements along all normal modes by means of energy and excited state gradient calculations at the ground-state equilibrium geometry. The method is valid for the IMDHO model for which the excited-state energy gradient along a given normal mode and corresponding origin shift are related in a very simple way.

```
#
# example010.inp
#
# TDDFT BHLYP Normal Mode Gradient Calculation
#
#   The keyword NMGrad invokes the normal mode gradient calculation
#
! RKS BHandHLYP TightSCF SV(P) NMGrad

%cis    NRoots 1
        triplets false
end

%rr
    # the nuclear Hessian must have been calculated before - for example by a
    # DFT calculation.
    HessName= "hexatriene.hess"

    states 1    # Perform energy-gradient calculations for the 1st
                # excited state.
    Tdnc 0.005  # Threshold for dimensionless displacements to be
                # included in the input file for spectral simulations
                # generated at the end of the program run.
                # By default Tdnc= 0.005

    ASAInput true # Generate the input file for spectra simulations
```

(continues on next page)

(continued from previous page)

```
 end

* xyz   0 1
  C    -0.003374    0.678229   0.00000
  H    -0.969173    1.203538   0.00000
  C     1.190547    1.505313   0.00000
  H     2.151896    0.972469   0.00000
  C     1.189404    2.852603   0.00000
  H     0.251463    3.423183   0.00000
  H     2.122793    3.426578   0.00000
  C     0.003374   -0.678229   0.00000
  H     0.969172   -1.203538   0.00000
  C    -1.190547   -1.505313   0.00000
  H    -2.151897   -0.972469   0.00000
  C    -1.189404   -2.852603   0.00000
  H    -0.251463   -3.423183    0.00000
  H    -2.122793   -3.426578    0.00000
*
```

In the ORCA run the TDDFT excited state gradient calculations are performed on top of a TDDFT calculation. Note, that the numbers of the excited-states which have to be included into analysis and input file for spectral simulations must be specified after the `States` keyword in the `%rr` block. They should also be consistent with the required number of roots in the `%tddft` block. The 1-^1B_u excited state appears to be the first root in the TDDFT calculation. Therefore, `NRoots=1` in the `%tddft` block, and `States=1` in the `%rr` block. One should also provide the name of the file containing the nuclear Hessian matrix via the `HessName` keyword in the `%rr` block. Here we used the `.hess` file obtained in a frequency calculation at the BHLYP/SV(P) level of theory.

After the ORCA calculation you will find in your directory a file called `example010.asa.inp` that is appropriate to be used together with the `orca_asa program` as defined in the preceding sections.

```
#
# example010.asa.inp
#
# ASA input
#
%sim
         model IMDHO
         method Heller
         AbsRange     5000.0,  100000.0
         NAbsPoints    0

         FlRange      5000.0,  100000.0
         NFlPoints     0

         RRPRange     5000.0,  100000.0
         NRRPPoints    0

         RRSRange        0.0,    4000.0
         NRRSPoints 4000

         RRS_FWHM 10.0


         AbsScaleMode Ext
         FlScaleMode Rel
         RamanOrder  0

         EnInput E0

         CAR   0.800
```

(continues on next page)

(continued from previous page)

```
end

%fit
         Fit false
         AbsFit false
         FlFit false
         RRPFit fsalse
         RRSFit false
         method Simplex
         WeightsAdjust true

         AbsRange        0.0,  10000000.0
         FlRange         0.0,  10000000.0
         RRPRange        0.0,  10000000.0
         RRSRange        0.0,  10000000.0
         AbsName ""
         FlName  ""

         ExpFlScaleMode  Rel
         ExpAbsScaleMode Rel

         CWAR  -1.000
         CWAF  -1.000

         NMaxFunc 100
         MWADRelTol= 1.000000e-004

         SFRRPSimStep= 1.000000e+002
         SFRRSSimStep 1.000000e+002
         FREQGStep 1.000000e+001
         FREQEStep 1.000000e+001
         E0Step 3.000000e+002
         SSStep 2.000000e+001
         TMStep 5.000000e-001
         GammaStep 5.000000e+001
         E0SDStep 5.000000e+001
         SDNCStep 4.000000e-001
end

$el_states
1
   1 42671.71   100.00    0.00     1.0725   3.3770  -0.0000

$vib_freq_gs
12
    1      359.709864
    2      456.925612
    3      974.521651
    4     1259.779018
    5     1356.134238
    6     1370.721341
    7     1476.878592
    8     1724.259894
    9     1804.572974
   10     3236.588264
   11     3244.034359
   12     3323.831066


$sdnc
12 1
```

(continues on next page)

(continued from previous page)

```
                    1
     1      -0.594359
     2       0.369227
     3      -0.132430
     4      -0.727616
     5       0.406841
     6      -0.105324
     7       0.177617
     8      -0.090105
     9      -1.412258
    10       0.048788
    11       0.021438
    12       0.008887
```

This input file can be used to construct theoretical absorption and rR spectra. **In order to compare experimental and theoretical rR spectra, it is necessary to use in both cases excitation energies that are approximately in resonance with the same vibrational transitions in the absorption spectrum. Therefore, in the case of the absorption spectrum with resolved or partially resolved vibrational structure it is necessary to modify the transition energies in the `%el_states` such that they coincide with the experimentally observed 0-0 vibrational peaks.** It is also desirable to roughly adjust homogeneous and, possibly, inhomogeneous linewidth parameters such that the experimental and calculated absorption spectra show similar slopes in the preresonance region (below the 0-0 transition). Then the assignment of experimental rR spectra can be done on the basis of comparison with the theoretical rR spectra calculated for the corresponding experimental excitation energies. For the sake of consistency and simplicity it is better to use those excitation energies which fall into the preresonace region and/or are in resonance with the 0-0 transition. **In the case of diffuse absorption spectra (i.e. those not showing resolved vibrational structure) it is also necessary to adjust the theoretical transition energies and linewidth parameters such that experimental and calculated positions of absorption maxima roughly coincide, and corresponding slopes below the maxima have a similar behavior.** According to above mentioned considerations one needs to modify the `%el_states` block in the file `example010.asa.inp`:

```
$el_states
1
   1 39808.0   150.00    0.00  1.0725   3.3770  -0.0000
```

The calculated absorption spectrum obtained by providing `AbsScaleMode= Rel`, `AbsRange= 39000`, `49000` and `NAbsPoints= 2000` is shown in Fig. 5.48. Upon comparison with the experimental spectrum one can notice that the BHLYP functional gives relatively small discrepancies with somewhat lower intensity in the low-frequency edge and larger intensity on the high-energy side of the spectrum. Besides, there is a noticeable mismatch in the separation between individual vibronic peaks which is due to overestimation of vibrational frequencies by the BHLYP functional (typically by $\approx 10\%$).

You can arbitrarily vary various normal coordinate displacements in `%sdnc` block within 10–30% of their values in order to observe modifications of the calculated spectrum. This will tell you how these parameters influence the spectrum and probably it will be possible to obtain better initial guesses for the fit. In the present example you will find that reduction of the absolute value of the displacement parameter corresponding to the ninth mode by $\approx 10\%$, and reduction of vibrational frequencies by $\approx 10\%$ can noticeably improve the spectral envelope. Such a quick analysis suggests that experimentally observed peaks in the absorption spectrum represent different vibrational transitions corresponding to a single electronically excited state rather than to different electronic excitations. This conclusion will be confirmed upon establishing the fact that the absorption and rR spectra can be successfully fitted based on the assumption of a single electronic transition.

Fig. 5.48: Experimental and calculated at the BHLYP/SV(P) and B3LYP/SV(P) levels of theory absorption (left panel) and rR spectra (right panel) corresponding to 1-$^1A_g \rightarrow$ 1-^1B_u transition in *trans*-1,3,5-hexatriene.

In order to calculate the rR spectrum for experimental excitation energies you need to specify its value through `RRSE` keyword in `%sim` block as well as possibly to modify the parameters related to the spectral range and linewidth of rR bands which are suitable for comparison with the experimental rR spectrum:

```
# excitation energies (cm**-1) for which rR spectra will be calculated:
   RRSE 39808

   # full width half maximum of Raman bands in rR spectra (cm**-1):
   RRS_FWHM  20

   RSRange   0, 4000  # spectral range for simulation of rR spectra (cm**-1)
   NRRSPoints  4000   # number of points to simulate rR spectra (cm**-1)

   # resonance Raman intensities will be calculated
   # for all vibrational states with excitation number
   # up to RamanOrder:
   RamanOrder 3
```

The calculated rR spectrum is shown in Fig. 5.48. In the input we have invoked the calculation of rR intensities for the transitions with up to 3 vibrational quanta in the final vibrational state (`RamanOrder = 3`). Make sure that the rR intensity pattern in the given spectral range does not change noticeably upon further increase of this parameter. Typically, the larger are the normal coordinate displacements the greater order of Raman scattering is required in the calculation to account for all the most intense transitions in the rR spectrum. The inclusion of vibrational transitions beyond the fundamentals is a particular feature of the `orca_asa` program.

Comparison of the calculated and experimental rR spectra (Fig. 5.48) mainly shows discrepancies in the values of the Raman shifts that are mainly related to the low accuracy of the vibrational frequencies obtained at the BHLYP level (typically overestimated by $\approx$ 10%). However, the intensity patterns of the calculated and experimental rR spectra show very nice agreement with experiment that is already sufficient to assign the experimental peaks to individual

vibrational transitions. This can be done upon examination of file `example010.asa.o3.rrs.39808.stk` which provides intensity, Raman shift, and specification for each vibrational transition. It is actually one of the most consistent procedures that enables one to identify different fundamental, overtone and combination bands in the experimentally observed rR spectrum. Such an assignment is a necessary prerequisite for the fit. The current example is relatively straightforward since the spectral region 1–1700 cm^{-1} is actually dominated by fundamental bands while the most intense overtone and combination transitions occur at higher frequencies. However, in many cases even the low-frequency spectral range is characterized by significant contributions from overtone and combination bands that sometimes are even more intense than fundamental transitions! Thus, quantum chemical calculations can greatly facilitate the assignment of experimental rR bands.

After having performed the assignment it is advisable to discard those modes from the analysis that are not involved in any of the experimentally observed fundamental, overtone, or combination rR bands with noticeable intensities. In the present example these are the modes 6, 8, 10–12 from the input file given above. For these modes it is implied that the fitted displacement parameters are zero. You will find that the calculated displacement values are rather small indeed. Also it is advisable to change the ground-state vibrational frequencies in the `$vib_freq_gs` block to their experimental values.

Below is the modified input file for the fit run:

```
#
# example010-01.asa.inp
#
# ASA input
#
%sim
         model IMDHO
         method Heller
end


%fit

    Fit true
    AbsFit true
    RRSFit true
    AbsExpName "hex-abs.dat"
    ExpAbsScaleMode Rel
    CWAR  5.0

    NMaxFunc  1000

    SDNCStep  0.5

end

$el_states
1
   1 39808.0   150.00    0.00    -0.8533  -3.3690  -0.0000

$el_states_c
1
   1     1        1       0           0        0       0


$vib_freq_gs
7
    1      354.0
    2      444.0
    3      934.0
    4     1192.0
    5     1290.0
    6     1403.0
```

(continues on next page)

(continued from previous page)

```
    7     1635.0



$sdnc
7 1
                    1
      1      -0.594359
      2       0.369227
      3      -0.132430
      4      -0.727616
      5       0.406841
      6       0.177617
      7      -1.412258

$sdnc_c
7 1
             1
      1        1
      2        2
      3        3
      4        4
      5        5
      6        6
      7        7

$rrs_exp
1
1 1
    Ex 39809.0
    NTr  11
    1
       int   10.0  1.0
       modes  1
       quanta 1;
    2
       int   5.0  1.0
       modes  2
       quanta 1;
    3
       int   1.5  1.0
       modes  3
       quanta 1;
    4
       int   21.0  1.0
       modes  4
       quanta 1;
    5
       int   7.5  1.0
       modes  5
       quanta 1;
    6
       int   2.0  1.0
       modes  6
       quanta 1;
    7
       int   46.0  1.0
       modes  7
       quanta 1;

    8
```

(continues on next page)

(continued from previous page)

```
      int  6.8 1.0
      modes  1, 7
      quanta 1, 1;

   9
      int  4.0 1.0
      modes  2, 7
      quanta 1, 1;

  10
      int  2.0 1.0
      modes  3, 7
      quanta 1, 1;

  11
      int 17.0 1.0
      modes  7
      quanta 2;
```

In addition to the experimental intensities of fundamental bands the input file also contains the information about some overtone and combination transitions. Note that it is not really necessary to include all of them them into the fit, in particular if some of the rR bands are strongly overlapping with each other.

Fitted normal coordinate displacements of the excited-state origin show nice agreement with the published values:

```
$sdnc
7 1
                    1
      1      -0.638244
      2       0.455355
      3      -0.229126
      4      -0.854357
      5       0.501219
      6       0.197679
      7      -1.292997
```

Note

- It is not really important to employ the BHLYP/SV(P) method in the frequency calculations in order to obtain the `.hess` file (this was merely done to be consistent with the TDDFT/BHLYP/SV(P) method for the excited-state model parameters calculation). The frequency calculations can for example be carried out at the BP86/TZVP or RI-SCS-MP2/TZVP level of theory. This will provide displacements pattern very similar to that of the BHLYP/SV(P) method, but much more accurate vibrational frequencies which will further facilitate the assignment of rR spectra (Fig. 5.48). However, such a procedure can be inconsistent if the two methods give noticeably different normal mode compositions and/or vibrational frequencies. From our experience it can lead to significant overestimation of the excited-state displacements for some low-frequency modes.
- It is known that predicted dimensionless normal coordinate displacements critically depend on the fraction of the "exact" Hartree-Fock exchange (EEX) included in hybrid functionals. In general no universal amount of EEX exists that provides a uniformly good description for all systems and states. Typically, for a given molecule either the BHLYP/TZVP (50% of EEX) or B3LYP/TZVP (20% of EEX) methods yields simulated spectra that compare very well with those from experiment if vibrational frequencies are appropriately scaled.

Important Notes about Proper Comparison of Experimental and Quantum Chemically Calculated Resonance Raman Spectra

In order to compare experimental and theoretical rR spectra, **it is necessary to use in both cases excitation energies that are approximately in resonance with the same vibrational transitions in the absorption spectrum**. Therefore, in the case of diffuse absorption spectra (i.e. those not showing resolved vibrational structure) one needs to adjust the transition energies and linewidth paramters in the `%el_states` block such that the envelopes of the experimental and theoretical spectra rouhgly coincide, and then to employ experimental values of excitation energies to construct theoretical rR spectra. Typically in the case of diffuse absorption spectra rR profiles are rather smooth. Therefore, even though excitation energies are not in resonance with the same vibrational transition in the absorption spectrum, the rR spectra are not expected to vary significantly in the case of such mismatch.

In the case of the absorption spectrum with resolved or partially resolved vibrational structure it is necessary to modify the transition energies in the `%el_states` block such that the calculated and experimentally observed 0-0 vibrational peaks coincide, and modify linewidth parameters so that the low-energy slopes in the calculated and experimental spectra have a similar behavior.

Consider a single-mode model system for which "experimental" and calculated absorption spectra are shown in Fig. 5.49.

Fig. 5.49: Experimental and theoretical absorption spectra for a single-mode model system. The calculated spectrum is adjusted such that the position of 0-0 peak coincide with the experimental one.

Comparison of the calculated and experimental spectra shows that some adjustment of the linewidth parameters is neceassy before construction of theoretical rR spectra. One can directly compare calculated and experimental rR spectra upon the excitation at 16200 cm^{-1} which is in resonance with the 0-0 vibronic band. However, it is not consistent to use experimental values of the excitation energy in the calculation of rR spectrum which is in resonance with one of the other vibronic bands since the separation between vibrational peaks in the experimental and calculated spectra is different whereby positions of the peaks in both spectra do not coincide. Instead **one should use the excitaition energy which corresponds to the same vibronic peak in the calculated absorption spectrum as in the experimental one**. Alternatively, one can adjust theoretical value of vibrational frequency such that positions of corresponding vibronic peaks in the spectra coincide, and then use experimental values of excitation energies for the calculation of rR spectra.

Example: Normal Mode Scan Calculations of Model Parameters for 1-$^1A_g \rightarrow$ 1-^1B_u Transition in *trans*-1,3,5-Hexatriene

If excited state gradients are not available (which is the case for many of the electronic structure methods supported by ORCA), you have to resort to a more laborious procedure – single point calculations at geometries that are displaced along the various normal modes of the system. This roughly corresponds to taking numerical derivatives – however, once this extra effort is invested more information can be obtained from the calculation than what would be possible from an analytic derivative calculation.

The present example illustrates the application of normal mode scan calculations for the evaluation of excited state harmonic parameters that are necessary to simulate optical spectra within the IMDHO model. This method can be applied with any method like CIS, CASSCF, MRCI or TD-DFT.

The reference wavefunctions for the multireference calculations reported below are of the state-averaged CASSCF (SA-CASSCF) type. The complete active space CAS(6,6) includes all 6 valence shell π-orbitals. The average is taken over the first four states which was found necessary in order to include the ground state and the strongly allowed 1-^1B_ustate.

```
#
# example011.inp
#
# CASSCF normal mode scan calculations
#

# first do single point RHF calculation
! RHF TZVP TightSCF

* xyz 0 1
 C       -0.002759    0.680006    0.000000
 H       -0.966741    1.204366    0.000000
 C        1.187413    1.500920    0.000000
 H        2.146702    0.969304    0.000000
 C        1.187413    2.850514    0.000000
 H        0.254386    3.420500    0.000000
 H        2.116263    3.422544    0.000000
 C        0.002759   -0.680006    0.000000
 H        0.966741   -1.204366    0.000000
 C       -1.187413   -1.500920    0.000000
 H       -2.146702   -0.969304    0.000000
 C       -1.187413   -2.850514    0.000000
 H       -0.254386   -3.420500    0.000000
 H       -2.116263   -3.422544    0.000000
*


# perform SA-CASSCF calculation upon appropriate rotation of MOs
$new_job
! TZVP TightSCF
```

(continues on next page)

(continued from previous page)

```
%scf
    rotate {23,27} end
end

%casscf
        nel 6
        norb  6
        mult  1
        nroots  4
end

* xyz 0 1
 C       -0.002759    0.680006    0.000000
 H       -0.966741    1.204366    0.000000
 C        1.187413    1.500920    0.000000
 H        2.146702    0.969304    0.000000
 C        1.187413    2.850514    0.000000
 H        0.254386    3.420500    0.000000
 H        2.116263    3.422544    0.000000
 C        0.002759   -0.680006    0.000000
 H        0.966741   -1.204366    0.000000
 C       -1.187413   -1.500920    0.000000
 H       -2.146702   -0.969304    0.000000
 C       -1.187413   -2.850514    0.000000
 H       -0.254386   -3.420500    0.000000
 H       -2.116263   -3.422544    0.000000
*


# do normal mode scan calculations
# to map CASSCF ground and excited-state PESs
$new_job
! TZVP TightSCF  NMScan

%casscf
        nel 6
        norb  6
        mult  1
        nroots  4
end

%rr
    HessName "hexatriene_bp86.hess"
    NMList 10,11,18,24,26,28,29,31,32
    NSteps 6
    FreqAlter true
    EnStep 0.0001
    State 3
end

* xyz 0 1
 C       -0.002759    0.680006    0.000000
 H       -0.966741    1.204366    0.000000
 C        1.187413    1.500920    0.000000
 H        2.146702    0.969304    0.000000
 C        1.187413    2.850514    0.000000
 H        0.254386    3.420500    0.000000
 H        2.116263    3.422544    0.000000
 C        0.002759   -0.680006    0.000000
 H        0.966741   -1.204366    0.000000
 C       -1.187413   -1.500920    0.000000
```

(continues on next page)

(continued from previous page)

```
 H       -2.146702   -0.969304    0.000000
 C       -1.187413   -2.850514    0.000000
 H       -0.254386   -3.420500    0.000000
 H       -2.116263   -3.422544    0.000000
*
```

The file containing the Hessian matrix (`"hexatriene_bp86.hess"`) was obtained from the BP86/TZVP frequency calculations. The keyword `NMList` provides the list of the normal modes to be scanned. These should be only the totally symmetric vibrations, since only they can be significant for absorption and resonance Raman spectra within the constraints of the IMDHO model. The `FreqAlter` flag indicates whether frequency alterations are assumed in the post-scan potential surface fit. The Parameter `EnStep` is used to select the appropriate step during the scan calculations. The value is chosen such that the average energy change (in Eh) in both directions is not less than this parameter.

5.17 Absorption and Fluorescence Bandshapes using ORCA_ASA

Please also consider using the more recent ORCA_ESD, described in Section *Excited State Dynamics*, to compute bandshapes.

Bandshape calculations are nontrivial but can be achieved with ORCA using the procedures described in section *Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program*. Starting from version 2.80, analytical TD-DFT gradients are available, which make these calculations quite fast and applicable without expert knowledge to larger molecules.

Note

- Functionals with somewhat more HF exchange produce better results and are not as prone to "ghost states" as GGA functionals unfortunately are!
- Calculations can be greatly sped up by the RI or RIJCOSX approximations!
- Analytic gradients for the (D) correction and hence for double-hybrid functionals are NOT available.

In a nutshell, let us look into the H_2CO molecule. First we generate some Hessian (e.g. BP86/SV(P)). Then we run the job that makes the input for the `orca_asa` program. For example, let us calculate the five lowest excited states:

```
! aug-cc-pVDZ BHandHLYP TightSCF NMGrad

%tddft nroots 5 end

# this is ASA-specific input
%rr    states 1,2,3,4,5
       HessName "Test-ASA-H2CO-freq.hess"
       ASAInput True
       end

*int 0 1
 C 0 0 0  0   0     0
 O 1 0 0  1.2 0     0
 H 1 2 0  1.1 120   0
 H 1 2 3  1.1 120 180
*
```

The ORCA run will produce a file `Test-ASA-H2CO.asa.inp` that is an input file for the program that generates various spectra. It is an ASCII file that is very similar in appearance to an ORCA input file:

```
#
# ASA input
#
%sim     model IMDHO
         method Heller

         AbsRange     25000.0,  100000.0
         NAbsPoints   1024

         FlRange      25000.0,  100000.0
         NFlPoints    1024

         RRPRange     5000.0,  100000.0
         NRRPPoints   1024

         RRSRange        0.0,    4000.0
         NRRSPoints   4000

         # Excitation energies (cm**-1) for which rR spectra will
         # be calculated. Here we choose all allowed transitions
         # and the position of the 0-0 band
         RRSE 58960, 66884, 66602

         # full width half maximum of Raman bands in rR spectra
         # (cm**-1):
         RRS_FWHM 10.0

         AbsScaleMode Ext
         FlScaleMode  Rel
         # RamanOrder=1 means only fundamentals. For 2 combination
         # bands and first overtones are also considered, for 3
         # one has second overtones etc.
         RamanOrder   1

         # E0 means the adiabatic excitation energy
         # EV would mean the vertical one. sprints vertical
         # excitations in the TD-DFT output but for the input into
         # the ASA program the adiabatic excitation energies are
         # estimated. A rigorous calculation would of course in-
         # volve excited state geometry optimization
         EnInput E0

         CAR   0.800
         end

# These are the calculated electronic states and transition moments
# Note that this is in the Franck-Condon approximation and thus
# the transition moments have been calculated vertically
$el_states
5
   1 32200.79   100.00    0.00    -0.0000   0.0000  -0.0000
   2 58960.05   100.00    0.00     0.0000  -0.4219   0.0000
   3 66884.30   100.00    0.00    -0.0000   0.4405   0.0000
   4 66602.64   100.00    0.00    -0.5217  -0.0000   0.0000
   5 72245.42   100.00    0.00     0.0000   0.0000   0.0000

# These are the calculated vibrational frequencies for the totally
# symmetric modes. These are the only ones that contribute. They
# correspond to x, H-C-H bending, C=O stretching and C-H stretching
# respectively
$vib_freq_gs
3
```

(continues on next page)

(continued from previous page)

```
    1      1462.948534
    2      1759.538581
    3      2812.815170

# These are the calculated dimensional displacements for all
# electronic states along all of the totally symmetric modes.
$sdnc
3 5
                    1          2          3          4          5
      1      -0.326244   0.241082  -0.132239   0.559635   0.292190
      2      -1.356209   0.529823   0.438703   0.416161   0.602301
      3      -0.183845   0.418242   0.267520   0.278880   0.231340
```

After setting `NAbsPoints` variable and spectral ranges in this file to the desired values, we invoke `orca_asa` as:

```
orca_asa Test-ASA-H2CO.asa.inp
```

This produces the following output:

```
******************
                                    * O R C A  A S A *
                                    ******************

                     --- A program for analysis of electronic spectra ---


Reading file: Test-ASA-H2CO.asa.inp ... done


         ***************************************************************
         *        GENERAL CHARACTERISTICS OF ELECTRONIC SPECTRA        *
         ***************************************************************

-------------------------------------------------------------------------------
  State      E0           EV        fosc    Stokes shift  Effective Stokes shift
          (cm**-1)     (cm**-1)               (cm**-1)           (cm**-1)
-------------------------------------------------------------------------------
    1:     30457.24   32200.79   0.000000      0.00                0.00
    2:     58424.56   58960.05   0.031879      0.00                0.00
    3:     66601.54   66884.30   0.039422      0.00                0.00
    4:     66111.80   66602.64   0.055063      0.00                0.00
    5:     71788.55   72245.42   0.000000      0.00                0.00


-------------------------------------------------------------------------------
↪---------------
                                BROADENING PARAMETETRS (cm**-1)
-------------------------------------------------------------------------------
↪---------------
                 Intrinsic                                    Effective
   State   -------------------------  ---------------------------------------------
↪------------
                                                      Sigma                       ␣
↪FWHM
            Gamma    Sigma    FWHM      ---------------------------   -------------
↪--------------
                                            0K       77K     298.15K      0K      ␣
↪77K     298.15K
-------------------------------------------------------------------------------
↪---------------
    1:     100.00     0.00   200.00        0.00      0.00      0.00    200.00   ␣
```

(continues on next page)

(continued from previous page)

```
↪200.00    200.00
    2:     100.00     0.00   200.00        0.00      0.00      0.00    200.00   ␣
↪200.00    200.00
    3:     100.00     0.00   200.00        0.00      0.00      0.00    200.00   ␣
↪200.00    200.00
    4:     100.00     0.00   200.00        0.00      0.00      0.00    200.00   ␣
↪200.00    200.00
    5:     100.00     0.00   200.00        0.00      0.00      0.00    200.00   ␣
↪200.00    200.00


Calculating absorption spectrum   ...
The maximum number of grid points ... 5840
Time for absorption               ...       9.569 sec (=   0.159 min)
Writing file: Test-ASA-H2CO.asa.abs.dat ... done
Writing file: Test-ASA-H2CO.asa.abs.as.dat ... done

Generating vibrational states up to the  1-th(st) order  ...   done
Total number of vibrational states                       ...      3

Calculating rR profiles for all vibrational states up to the 1-th order
State 1 ...
The maximum number of grid points ... 6820
Resonance Raman profile is done
State 2 ...
The maximum number of grid points ... 6820
Resonance Raman profile is done
State 3 ...
The maximum number of grid points ... 6820
Resonance Raman profile is done
Writing file: Test-ASA-H2CO.asa.o1.dat... done
Writing file: Test-ASA-H2CO.asa.o1.info... done

Calculating rR spectra involving vibrational states up to the 1-th(st) order
State    1 ... done
State    2 ... done
State    3 ... done

Writing file: Test-ASA-H2CO.asa.o1.rrs.58960.dat      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.58960.stk      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.66884.dat      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.66884.stk      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.66602.dat      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.66602.stk      ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.58960.dat   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.58960.stk   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.66884.dat   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.66884.stk   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.66602.dat   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.as.66602.stk   ... done
Writing file: Test-ASA-H2CO.asa.o1.rrs.all.xyz.dat    ... done

TOTAL RUN TIME: 0 days 0 hours 1 minutes 17 seconds 850 msec
```

The computed vibrationally resolved absorption spectrum is plotted as shown in Figure Fig. 5.50.

Fig. 5.50: The computed vibrationally resolved absorption spectrum of the H_2CO molecule

The computed fluorescence spectrum of the lowest energy peak is plotted as shown in Figure Fig. 5.51. This peak corresponds to S2. Although it is not realistic, it is sufficient for illustrative purposes.

Fig. 5.51: The computed fluorescence spectrum of the lowest energy peak of the H_2CO molecule

The computed Resonance Raman (rR) excitation profiles of the three totally symmetric vibrational modes are plotted as shown in Figure Fig. 5.52.

Fig. 5.52: The computed Resonance Raman excitation profiles of the three totally symmetric vibrational modes of the H_2CO molecule

As might be expected, the dominant enhancement occurs under the main peaks for the C=O stretching vibration. Higher energy excitations particularly enhance the C-H vibrations. The computed rR spectra at the vertical excitation energies are provided in Figure Fig. 5.53.

Fig. 5.53: The computed Resonance Raman spectra at the vertical excitation energies of the H_2CO molecule

In this toy example, the dominant mode is the C=O stretching, and the spectra look similar for all excitation wavelengths. However, electronically excited states are mostly of different natures, yielding drastically different rR spectra. Thus, rR spectra serve as powerful fingerprints of the electronic excitation being studied. This is also true even if the vibrational structure of the absorption band is not resolved, which is usually the case for large molecules.

The `orca_asa` program is much more powerful than described in this section. Please refer to section *Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program* for a full description of its features. The `orca_asa` program can also be interfaced to other electronic structure codes that deliver excited state gradients and can be used to fit experimental data. It is thus a tool for experimentalists and theoreticians at the same time!

5.18 Vibrational Spectroscopy

The fundamental usage and several examples for computing vibrational frequencies in ORCA can be found here: *Vibrational Frequencies*. The following section is more spectroscopy focused.

5.18.1 IR Spectra

****There were significant changes in the IR printing after ORCA 4.2.1!****

IR spectral intensities are calculated automatically in frequency runs. Thus, there is nothing control by the user. Consider the following job:

```
! OPT FREQ BP86 def2-SVP

*xyz 0 1
 O    0.000000    0.000000    0.611880
 C    0.000000    0.000000   -0.596849
 H    0.952616    0.000000   -1.209311
 H   -0.952616    0.000000   -1.209311
*
```

which gives the following output:

```
-----------
IR SPECTRUM
-----------

 Mode   freq       eps      Int      T**2         TX        TY        TZ
       cm**-1   L/(mol*cm) km/mol    a.u.
-----------------------------------------------------------------------------
  6:   1146.68   0.000341    1.73  0.000093  (-0.000000 -0.009640  0.000000)
  7:   1224.67   0.002004   10.13  0.000511  ( 0.022596  0.000000  0.000000)
  8:   1485.77   0.001002    5.07  0.000211  ( 0.000000 -0.000000  0.014510)
  9:   1806.49   0.020286  102.51  0.003504  ( 0.000000 -0.000000  0.059197)
 10:   2769.13   0.014010   70.80  0.001579  ( 0.000000  0.000000  0.039734)
 11:   2812.52   0.039321  198.71  0.004363  ( 0.066052 -0.000000 -0.000000)
```

The first column ('Mode') labels vibrational modes that increase in frequency from top to bottom." The next column provides vibrational frequencies. The molar absorption coefficient ε of each mode is listed in the "eps" column. This quantity is directly proportional to the intensity of a given fundamental in an IR spectrum, and thus it is used by the `orca_mapspc` utility program as the IR intensity.

The values under "Int" are the integrated absorption coefficient[1], and the "T**2" column lists the norm of the transition dipole derivatives, already including the vibrational part.

To obtain a plot of the spectrum, the `orca_mapspc` utility can be run calling the output file as:

```
orca_mapspc Test-Freq-H2CO.out ir -w25
```

or calling the Hessian file as:

```
orca_mapspc Test-Freq-H2CO.hess ir -w25
```

The basic options of `orca_mapspc` are listed below:

```
-w  : a value for the linewidth (gaussian shape, fwhm)
-x0 : start value of the spectrum in cm**-1
-x1 : end value of the spectrum in cm**-1
-n  : number of points to use
```

[1] Explained in more detail by Neugbauer [679]

To see its options in detail, call `orca_mapspc` without any input. The above `orca_mapspc` runs of the H_2CO molecule provide `Test-NumFreq-H2CO.out.ir.dat` file that contains intensity and wavenumber columns. Therefore, this file can serve as input for any graph plotting program. The plot of the computed IR spectrum of the H_2CO molecule obtained with the above ORCA run is as given in Figure Fig. 5.54.

Fig. 5.54: The predicted IR spectrum of the H_2CO molecule plotted using the file generated by the `orca_mapspc` tool.

5.18.2 Overtones, Combination Bands and Near IR Spectra via NEARIR

Overtones and combination bands can also be incorporated to the computed IR or Near IR spectrum for completeness. The intensities of these bands are strongly dependent on anharmonic effects. ORCA can include these effects by means of the VPT2 approach [673]. The full cubic force field, anharmonic corrections to overtones and combination bands, and a broad range of methods are available in the `orca_vpt2` module (see section *Anharmonic Analysis and Vibrational Corrections using VPT2/GVPT2 and orca_vpt2*).

In particular, the `NEARIR` keyword calls a simpler semidiagonal approach, including only two modes (i and j, also refered as 2MR-QFF in [674, 675]) and force constants up to cubic order (k_{iij}, k_{iji} and k_{iii}). For now, only the intensities are corrected for anharmonic effects - **frequencies are not.**

Overtones and Combination Bands

Since the calculation of these terms scale with N^2_{modes}, it can quickly become too expensive, thus we use by default the semiempirical GFN2-xTB [320] to compute the energies and dipole moments necessary to the higher order derivatives (which can be changed later). To request this, simply add `!NEARIR` in the main input. An example input for computing the fundamentals of toluene using B2PLYP double-hybrid functional and for computing the anharmonics using XTB is as follows:

```
! TightOPT NumFREQ RI-B2PLYP def2-TZVP def2-TZVP/C RIJCOSX NEARIR

*xyzfile 0 1 toluene.xyz
```

Note

These anharmonic corrections are very sensitive to the geometry. Therefore, perform a conservative geometry optimization (at least `TightOPT`) whenever possible.

In the output, the characteristics of the regular IR spectrum are printed first. Then, the characteristics of overtones and combination bands are provided similarly to the fundamentals, as follows:

```
-------------------------------
OVERTONES AND COMBINATION BANDS
-------------------------------

Mode      freq       eps       Int      T**2        TX        TY        TZ
         cm**-1   L/(mol*cm)  km/mol    a.u.
-----------------------------------------------------------------------------
6+6:      64.71    0.000994    5.02   0.004792  (-0.009428 -0.066232  0.017796)
6+7:      241.83   0.000022    0.11   0.000028  (-0.005268  0.000255  0.000638)
6+8:      375.36   0.000048    0.24   0.000040  (-0.000740  0.001917  0.006007)
6+9:      442.49   0.000000    0.00   0.000000  ( 0.000010  0.000001  0.000001)
6+10:     506.37   0.000003    0.01   0.000002  ( 0.001078 -0.000061  0.000799)
(...)
```

The "Mode" column shows the overtones, such as 6+6, and combination bands, such as 6+7 and 6+8. These new quantities are automatically detected and incorporated in the IR spectrum when the output file is called with the `orca_mapspc` utility as follows:

```
orca_mapspc toluene-nearir.out ir -w25
```

From the file `orca_mapspc` provided, the IR spectrum can be plotted as shown in Figure Fig. 5.55.

Fig. 5.55: Calculated and experimental infrared spectrum of toluene in gas phase. While the red plot includes only the fundamentals, the blue plot includes also overtones and combination bands. The grey dashed plot is the experimental gas-phase spectrum obtained from the NIST database. The theoretical frequencies were scaled following literature values [676]

"Benzene fingers", i.e., overtones and combination bands of the ring, are recovered in the computed spectrum. Note that the frequencies were scaled using literature values [676], and are not yet corrected using VPT2.

Near IR Spectra

Let us simulate near IR spectrum of methanol in CCl_4, as published by Bec and Huck [677], using B3LYP for fundamentals, XTB for overtones, and CPCM for solvation. The input is as follows:

```
! TightOPT FREQ B3LYP def2-TZVP RIJCOSX NEARIR CPCM(CCl4)

*xyz 0 1
 O      0.39517      4.38840     -0.00683
 C     -0.50818      3.29837      0.00221
 H     -0.11943      5.18771      0.19752
 H      0.03977      2.38083     -0.22470
 H     -1.27919      3.45664     -0.75583
 H     -0.96616      3.21170      0.99058
*
```

Calling the output with `orca_mapspc` by setting final point to about 8000 cm^{-1} in order to extend the spectrum to the near IR region, i.e.,

```
orca_mapspc toluene-nearir.out ir -w25 -x18000
```

one can simulate the spectrum from the generated "toluene-nearir.dat" file. As seen in Figure Fig. 5.56 the computed spectrum plotted with scaled computational frequencies (not yet corrected using VPT2) according to [676] agrees reasonably well with the experimental spectrum.

Fig. 5.56: Calculated and experimental near IR spectrum of methanol in CCl_4. The blue plot is for overtones; the red plot is for combination bands; and the grey dashed plot is the experimental spectrum. Theoretical frequencies were scaled according to literature values [676].

Using Other Methods for the VPT2 Correction

To compute overtones with the method chosen for the calculation of the fundamentals, one needs only to set `XTBVPT2` option in the `%freq` block to false, i.e.,

```
%freq XTBVPT2 False end
```

To set a different method for the calculation of overtones and combinations than used for the calculation of fundamentals, one needs first to perform a frequency calculation, then call the resulting Hessian file in `%geom` block, and activate the `PRINTTHERMOCHEM` flag (see section *Thermochemistry* for details), i.e.,

```
! BP86 def2-TZVP NEARIR CPCM(CCl4) PRINTTHERMOCHEM

%geom INHESSNAME "methanol.hess" end
%freq XTBVPT2 False end

*xyzfile 0 1 methanol_opt.xyz
```

In this example, the fundamental modes are read from the "methanol.hess" file, but the anharmonics and intensities of the overtones and combinations are computed using BP86. Any combination of methods, such as B3LYP/BP86 and B2PLYP/AM1, is allowed. Note that this description is an approximation to full VPT2 or GVPT2. For a more complete treatment, see the VPT2 module described in section *Anharmonic Analysis and Vibrational Corrections using VPT2/GVPT2 and orca_vpt2*.

By default, a step size of 0.5 in dimensionless normal mode unit is used during the numerical calculations. This can be changed by setting `DELQ` in the `%freq` block:

```
%freq
    XTBVPT2 False
    DELQ    0.1
end
```

The complete list of options related to VPT2 and in general frequency calculations can be found in Sec. *Vibrational Frequencies*.

5.18.3 Vibrational Circular Dichroism (VCD) Spectra

Vibrational circular dichroism spectrum calculations are implemented analytically at the SCF (HF or DFT) level following the derivation of Weigend and coworkers [678]. The basic usage is shown in the following example:

```
# AnFreq + doVCD triggers the VCD calculation
! AnFreq B3LYP def2-SVP

%freq
  doVCD true
end

*xyz 0 1
  C      1.231429   -0.226472   -0.084960
  C     -0.061893    0.507641    0.134338
  C     -1.358912   -0.147897    0.084831
  O     -0.902881    0.641038   -0.969176
  H      1.070541   -1.118875   -0.689778
  H      1.672013   -0.522768    0.869009
  H      1.946503    0.413187   -0.605194
  H      0.017832    1.411161    0.734623
  H     -1.417896   -1.212878   -0.118068
  H     -2.196737    0.255864    0.644375
*
```

Note that in addition to the Hessian, the VCD calculation requires the magnetic field response using GIAOs and the electric field response with the field origin placed at (0,0,0). The latter matches the hard-coded magnetic field gauge origin in the GIAO case and is necessary to ensure gauge-invariance of the results. ORCA does all of this automatically but it means that if VCD is requested together with electric and/or magnetic properties in the same job, the field origins cannot be changed.

Other keywords that influence the VCD calculation include `GIAO_1el` and `GIAO_2el` in `%eprnmr` and `CutOffFreq` in `%freq`. Note also that VCD cannot be computed with `NumFreq`.

5.18.4 Raman Spectra

Starting with ORCA 6.1, calculation of Raman intensities has been implemented using analytical geometric derivatives of the polarizability tensor and is now compatible with analytical frequency runs for SCF methods (HF, DFT). If a frequency run (`!Freq`) is combined with a polarizability calculation, the Raman spectrum will be automatically calculated.

Consider the following example:

```
! OPT RHF STO-3G TightSCF SmallPrint Freq

%elprop Polar 1 end

*xyz 0 1
 C    0.000000    0.000000   -0.533905
 O    0.000000    0.000000    0.682807
 H    0.000000    0.926563   -1.129511
 H    0.000000   -0.926563   -1.129511
*
```

The output provides the Raman scattering activity (in $Å^4$/AMU) [679] and the Raman depolarization ratio of each mode:

```
--------------
RAMAN SPECTRUM
--------------

 Mode    freq (cm**-1)   Activity   Depolarization
-----------------------------------------------------------------
   6:      1278.43      0.007371      0.750000
   7:      1397.30      3.043852      0.750000
   8:      1767.07     16.391442      0.707250
   9:      2099.28      6.704492      0.075590
  10:      3498.60     38.681286      0.186510
  11:      3645.53     24.527417      0.750000
```

The ORCA run also generates a `.hess` file that includes polarizability derivatives and Raman activities. The effect of isotope substitution on the Raman activities can be computed using the `.hess` file.

As in the IR spectrum case, `orca_mapspc` provides a `.dat` file for plotting the computed Raman spectrum:

```
orca_mapspc Test-NumFreq-H2CO.out raman -w50
```

The Raman spectrum of H_2CO plotted by using the corresponding `.dat` file is as given in Figure Fig. 5.57.

Fig. 5.57: Calculated Raman spectrum of H_2CO at the STO-3G level plotted using the `.dat` generated by the `orca_mapspc` utility from numerical frequencies and Raman activities.

It is worth noting that Raman scattering activity S_i of each mode i is related to but not directly equal to the Raman intensity I_i of the corresponding mode, which is dependent on the excitation line ν_0 of the laser used in the Raman measurement(for Nd:YAG laser: ν_0 = 1064 nm = 9398.5 cm^{-1}). To obtain significantly better agreement between experimental and simulated Raman spectra, I_i of each mode needs to be computed with the following formula:

$$I_i = \frac{f(\nu_0 - \nu_i)^4 S_i}{\nu_i[1 - \exp(-hc\nu_i/kT)]}$$

where f is a normalization constant common for all modes; h, c, k, and T are Planck's constant, speed of light, Boltzmann's constant, and temperature, respectively.

Note

- It is still possible to calculate Raman intensities using numerical derivatives of the analytical polarizability, if the combination of `!NumFreq` and `%elprop polar 1 end` is specified. This is currently the only option for derivatives of the dynamical polarizability. In the static polarizability case, this is significantly less efficient and requires tighter accuracy settings than the analytical derivative.
- The analytical Raman intensities calculation is also implemented for implicit solvation, specified e.g. using `!CPCM(solvent)` in the simple input.
- The analytical Raman intensities make use of RI-J and/or COSX approximations in case these are used for the SCF. As a note of caution, we have observed that the error of the RI-J approximation may become larger with the default auxiliary basis in case you use a basis set with diffuse functions (e.g. `def2-SVPD`).

5.18.5 Resonance Raman Spectra

Resonance Raman spectra (NRVS) and excitation profiles can be predicted or fitted using the procedures described in section *Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program*. An example for obtaining the necessary orca_asa input is described in section *Absorption and Fluorescence Bandshapes using ORCA_ASA*.

5.18.6 NRVS Spectra

The details of the theory and implementation of NRVS spectrum are as described in ref. [680, 681]. The NRVS spectrum of *iron-containing molecules* can be simply calculated calling `.hess` file of a previous frequency calculation with the `orca_vib` utility. The output file of this utility can then be called with `orca_mapspc` utility to produce a `.dat` file for plotting the spectrum:

```
orca_vib MyJob.hess > MyJob.vib.out
orca_mapspc MyJob.vib.out NRVS
```

For a the ferric-azide complex [680], the computed and experimental NRVS spectra are provided in Figure Fig. 5.58.

Fig. 5.58: Experimental (a, black curve), fitted (a, red) and simulated (b) NRVS spectrum of the Fe(III)-azide complex obtained at the BP86/TZVP level (T = 20 K). Bar graphs represent the corresponding intensities of the individual vibrational transitions. The blue curve represents the fitted spectrum with a background line removed.

As for the calculation of resonance Raman spectra described in section *Simulation and Fit of Vibronic Structure in Electronic Spectra, Resonance Raman Excitation Profiles and Spectra with the orca_asa Program*, the DFT estimations are usually excellent starting points for least-square refinements.

Below we describe the procedure for computing such NRVS spectra on the $Fe(SH)_4^{1-}$ complex with the BP86 functional, which typically provides good NRVS spectra. One needs first to optimize the geometry of the complex and compute its vibrational structure:

```
! OPT FREQ BP86 def2-TZVP TightSCF SmallPrint

*xyz -1 6
 Fe     -0.115452      0.019090     -0.059506
 S      -0.115452      1.781846      1.465006
 S      -0.115452     -1.743665      1.462801
 S      -1.908178     -0.072782     -1.518702
 S       1.560523      0.154286     -1.656664
 H       0.410700      2.760449      0.687716
 H      -0.674147     -2.708278      0.690223
 H      -2.905212      0.345589     -0.699907
 H       2.647892     -0.211681     -0.932926
*
```

Now run the `orca_vib` utility on the `.hess` file generated by this job to obtain an output file that can be used with `orca_mapspc` utility:

```
orca_vib Test-FeIIISH4-NumFreq.hess > Test-FeIIISH4-NumFreq.out
orca_mapspc Test-FeIIISH4-NumFreq.out NRVS
```

This `orca_mapspc` run generates `Test-FeIIISH4-NumFreq.nrvs.dat` file in the xy-format. This file contains phonon energy (x, in cm^{-1}) and NRVS intensity (y, in atomic units) and thus can be directly used for visualizing the spectrum.

The corresponding NRVS spectrum is given in Figure Fig. 5.59 together with the computational IR spectrum on the same frequency scale. NRVS reports the Doppler broadening of the Moessbauer signal due to resonant scattering of phonons (vibrations) dominated by the movements of Fe nuclei. This is a valuable addition to IR spectrum where the modes have very small intensities.

Fig. 5.59: (a) Theoretical IR spectrum of $Fe(SH)_4^{1-}$ and arrow-pictures of the highest intensity modes around the peak maxima. (b) The corresponding NRVS scattering spectrum.

5.18.7 Animation of Vibrational Modes

For describing how to animate vibrational modes and generate their "arrow-pictures", let us perform a frequency calculation on H_2CO:

```
! OPT FREQ RHF STO-3G

*xyz 0 1
 C    0.000000    0.000000   -0.533905
 O    0.000000    0.000000    0.682807
 H    0.000000    0.926563   -1.129511
 H    0.000000   -0.926563   -1.129511
*
```

The output of this job provides vibrational characteristics:

```
Mode      freq       eps       Int    T**2         TX        TY        TZ
         cm**-1   L/(mol*cm) km/mol    a.u.
-----------------------------------------------------------------------------
6:       1278.37   0.001222    6.18  0.000298  (-0.017272  0.000000  0.000000)
7:       1397.26   0.005844   29.53  0.001305  ( 0.000000  0.036128  0.000000)
8:       1767.02   0.000828    4.18  0.000146  (-0.000000  0.000000 -0.012089)
9:       2099.24   0.001668    8.43  0.000248  ( 0.000000 -0.000000  0.015749)
10:      3498.54   0.000356    1.80  0.000032  ( 0.000000 -0.000000 -0.005636)
11:      3645.47   0.003922   19.82  0.000336  (-0.000000  0.018322  0.000000)
```

This output can be directly opened with `ChemCraft` to visualize normal modes of H_2CO and to extract their arrow-pictures representing the direction of nuclear movements as shown in Figure Fig. 5.60. As an example, one can infer from this figure that the 1397 cm^{-1} mode corresponds to a rocking vibration.

Fig. 5.60: Normal modes of H_2CO with arrows indicating magnitude and direction of nuclear motions and the associated vibrational frequencies in cm^{-1} obtained from `ORCA` output file through the use of `ChemCraft`

In order to animate vibrational modes and to create their "arrow-pictures" by using free program packages like gOpenMol, the small utility program orca_pltvib can be used. This utility program generates a series of files from an ORCA output file of a frequency run, which can be openned with molecular visualization programs. The usage of orca_pltvib is as follows:

```
orca_pltvib Test-FREQ-H2CO.out [list of vibrations or all]
```

For example, let us animate the 1397 cm^{-1} mode labeled as 7:

```
orca_pltvib Test-FREQ-H2CO.out 7
```

This call will generate the Test-FREQ-H2CO.out.v007.xyz file. Open gOpenMol and read this file (Import->coords) in Xmol format. Then, go to the Trajectory->Main menu and import again the file in Xmol format. Now you are able to animate the mode. In order to generate a printable picture, press Dismiss and then type lulVectorDemo {4 0.1 black} into the gOpenMol command line window. The generated picture (see Figure Fig. 5.61) demonstrates that this mode corresponds to a rocking vibration.

Fig. 5.61: The 1397 cm^{-1} mode of the H_2CO molecule as obtained from the interface of ORCA to gOpenMol and the orca_pltvib tool to create the animation file.

The appearence of the arrows can also be modified as described in the web tutorial of gOpenMol.

5.18.8 Isotope Shifts

The calculated isotope shifts greatly aid in the identification of vibrations, the interpretation of experiments, and the assessment of the reliability of the calculated vibrational normal modes. It would be a very bad practice to recalculate the Hessian for investigating isotope shift since Hessian calculations are typically expensive, and the Hessian itself is independent of the masses. Below we describe how to find the isotope effect without recomputing the Hessian.

Let us suppose that you have calculated a Hessian as in the example discussed above, and you want to predict the effect of ^{18}O substitution. In this case you can use the small utility program `orca_vib`. First of all you need to edit the masses given in the `.hess` file by hand. For the example given above, the `.hess` file is as follows:

```
$orca_hessian_file
.......................
$hessian
12
... the cartesian Hessian in Eh/bohr**"

$vibrational_frequencies
12

...the vibrational frequencies (in cm-1) as in the output

$normal_modes
12 12
... the vibrational normal modes in Cartesian displacements
#
# The atoms: label  mass x  y  z
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#   Here we have changed 15.999 for oxygen into
#   18.0  in order to see the oxygen 18 effects
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$atoms
4
 C     12.0110      0.000000     0.000000    -1.149571
 O     18.0000     -0.000000    -0.000000     1.149695
 H      1.0080     -0.000000     1.750696    -2.275041
 H      1.0080     -0.000000    -1.750696    -2.275041

$actual_temperature
  0.000000

$dipole_derivatives
12

... the dipole derivatives (Cartesian displacements)
#
# The IR spectrum
#  wavenumber T**2 TX TY  TY
#
$ir_spectrum
12
... the IR intensities
```

After changing the mass of O from 15.999 to 18.0 as shown above, let us call:

```
orca_vib Test-FREQ-H2CO.hess
```

This will recompute vibrational frequencies in the presence of ^{18}O. Let us compare vibrational frequencies in the output of this run with the original frequencies in cm^{-1}:

```
  Mode       H2C16O    H2CO18O    Shift
----------------------------------------
```

(continues on next page)

(continued from previous page)

```
   6:        1284.36   1282.82    -1.54
   7:        1397.40   1391.74    -5.66
   8:        1766.60   1751.62   -14.98
   9:        2099.20   2061.49   -37.71
  10:        3499.11   3499.02    -0.09
  11:        3645.24   3645.24     0.00
```

Another way to analyze isotope shifts is to plot the two predicted spectra and then subtract one from the other. This will produce derivative-shaped peaks with zero crossings at the positions of the isotope-sensitive modes.

Note

In the presence of point charges and/or an external electric field, the translational and rotational symmetries of the system may be broken. In such cases, you may prefer NOT to project out the translational and rotational degrees of freedom of the Hessian. This can be achieved as:

```
orca_vib Test-FREQ-H2CO.hess -noproj
```

5.19 Anharmonic Analysis and Vibrational Corrections using VPT2/GVPT2 and `orca_vpt2`

Building upon (analytical) harmonic calculations of the Hessian, it is possible to calculate the cubic plus semi-quartic force field as well as higher-order property derivatives. For this purpose, the VPT2 module will compute the Hessian and then generate two displaced geometries for each degree of freedom and for each displacement another Hessian (and another property in case of vibrational corrections) will be computed. These are required for an anharmonic analysis according to second-order vibrational perturbation theory. So overall, using VPT2 is costly due to the number of calculations required for the numerical derivatives and is very sensitive to numerical noise due to convergence, approximations and other settings. The VPT2 calculation can be initiated either via the simple input command `!VPT2` or via the VPT2 keyword in the `%vpt2` block. Finer control can be achieved through the `%VPT2` block, as exemplified in this analysis of water.

```
# VPT2 Analysis of H2O
!RHF def2-SVP ExtremeSCF VPT2

%vpt2
    VPT2                On      # do a VPT2 analysis, same as !VPT2 (see above)
    AnharmDisp          0.05    # anharmonic displacement factor, default is 0.05
    HessianCutoff       1e-12   # cut-off for Hessian elements, default is $10^{-
↪10}$
    PrintLevel          1       # VPT2 print level [1, 2, 3, 4]
    MinimiseOrcaPrint   True    # Minimises the remaining orca output
end

%method
    Z_Tol               1e-14
end

* xyz 0 1
  O   0.00000000000000      0.06256176106279      0.06256176106280
  H   0.00000000000000     -0.06185639479702      0.99929463373422
  H   0.00000000000000      0.99929463373424     -0.06185639479703
*
```

After the analysis, a `<basename>.vpt2` file should be present in the working directory. Within that file all the force field and property derivatives are saved. It is used as an input for the `orca_vpt2` programme which is

called automatically after the initial displacement calculations. The programme can also be called separately with the command `orca_vpt2 <basename>.vpt2`.

Note

A few remarks about VPT2 calculations:

- A VPT2 starting geometry **should always be tightly converged**. For small molecules the `!TightOPT` option is not good enough ! Depending on your structure, you might want to experiment with the `TolE`, `TolRMSG` and `TolMaxG` keywords of the `%geom` block.
- Similarly, a well converged SCF is required. The use of the `ExtremeSCF` keyword or at least `VeryTightSCF` is recommended.
- The CP-SCF equations should be converged to at least 10^{-12} (modified via the `Z_Tol` setting in the `%method` block.
- For DFT calculations, tight grids like `DEFGRID3` are strongly recommended.
- **Linear molecules are not supported yet**
- Currently, only methods for which analytical Hessians are available are supported. Furthermore, **VPT2 calculations with DFT functionals which do not provide analytical Hessians cannot be carried out**.
- By default, updated atomic masses are used to generate the semi-quartic force field (see *Mass dependencies*). The masses are also printed in the `<basename>.vpt2` file
- A VPT2 analysis can be repeated on a previous calculation by running `orca_vpt2 <basename>.vpt2`.
- VPT2 does have limited restart capabilities. If the directory in which the VPT2 run is carried out already contains `<basename>.hess` or `<basename>_eprnmr.property.txt` files, the program will skip these points and use the information provided in the files.

VPT2 provides a vibrational analysis and thus access to :

- mean and mean square displacement expectation values
- centrifugal distortion constants
- Watson's symmetrically and asymmetrically reduced Hamiltonian parameters
- anharmonic constants
- Fermi resonance analysis
- rotational and vibrational-rotational constants
- fundamental transition (anharmonic frequencies)
- zero-point ro-vibrational energies
- overtones and combination bands with intensities (in contrast to NEARIR with full VPT2/GVBT2 treatment)
- dedicated file interface for codes like SPCAT

If the computed data should be used for the simulation of spectra with codes like SPCAT, ORCA can provide a dedicated file that can serve as a basis for an input. This is triggered in the `%output` block when a VPT2 calculation is run:

```
%output
 Pickettname "pickett.txt"
end
```

This way, ORCA will generate a file called `pickett.txt` that contains the computed data and templates for `.var` which can be modified to serve as input for codes like SPCAT. Please note that this feature is still being refined and extended.

5.19.1 Vibrational corrections of molecular properties using VPT2

Using VPT2 it is also possible to compute zero-point vibrational corrections to molecular properties. Currently, this is available for NMR chemical shieldings, spin-spin coupling constants, g- and A-tensors and requires two successive calculations. The first calculation is a VPT2 calculation just as shown above (`<basename>.inp`) that contains the `VPT2` command and the level of theory at which the Hessians are computed. The second calculation (let's call it `<basename>_Prop.inp` will compute the property derivatives with a final call to VPT2. In order for this to work, the property derivative calculation needs to read the `<basename>.hess` and `<basename>.vpt2` file from the forcefield calculation. This is done by specifying the `%geom inhess read` with the command `inhessname "<basename>.hess"`. This scheme is necessary as properties other than energies, geometries or frequencies often require specialized methods and basis sets. For the numerical calculation of the force field and property derivatives different stepsizes can be used by specifying `AnharmDisp` and `PropDisp` in the VPT2 input block. The defaults are 0.05, and after the calculation, the displaced geometries are stored in files named `myjob_DH001.xyz` and `myjob_DP001.xyz` etc.

A typical example for calculating the vibrational correction to the ^{13}C NMR chemical shifts of methanol with a B3LYP/def2-TZVP anharmonic forcefield and TPSS/pcSseg-2 shielding tensors would look like the following: the standard input file, in our case `vpt2_methanol_FF.inp` with the level of theory for the Hessian and the VPT2 input block :

```
!B3LYP D3 def2-TZVP def2/J def2/JK DEFGRID3 ExtremeSCF VPT2

%method
    Z_Tol  1e-12
end

* xyz 0 1
C  -1.09849212248373      0.14540972773089     -0.00000275092982
O   0.32138758531316      0.08706714755687     -0.00001212477411
H   0.66732439683790      0.98510769198508      0.00001819506998
H  -1.45583606337199     -0.88374271593276      0.00000595999622
H  -1.49206267729630      0.64725244577978      0.89143349761200
H  -1.49208273899904      0.64724452288014     -0.89144277697426
*
```

and the next input file, say `vpt2_methanol_NMR.inp` with the same geometry and the level of theory for the shielding tensor will look like this:

```
!TPSS pcSseg-2 Autoaux ExtremeSCF NMR

%geom inhess read
 inhessname "vpt2_methanol_FF.hess"
end

%vpt2
  VPT2          on
  AvgProp       NMR
  HessianCutoff 1e-12
end

%method
  Z_Tol         1e-12
end

* xyz 0 1
C  -1.09849212248373      0.14540972773089     -0.00000275092982
O   0.32138758531316      0.08706714755687     -0.00001212477411
H   0.66732439683790      0.98510769198508      0.00001819506998
H  -1.45583606337199     -0.88374271593276      0.00000595999622
H  -1.49206267729630      0.64725244577978      0.89143349761200
H  -1.49208273899904      0.64724452288014     -0.89144277697426
*
```

Running ORCA successively on both of these input files in the same directory will yield an output that contains the zero-point vibrational corrections to the shielding tensors for each atom. For Atom 0, which is the carbon in methanol, it looks like this:

```
-----
 Vibrationally averaged isotropic shieldings
 ----

Atom 0 :

 mode    <q>       <q2>      dS/dQ    d2S/dQ2
-----------------------------------------------
   0  -0.000017   0.202578  -0.000089  -5.922644
   1  -0.034052   0.057707   8.269988  -5.666515
   2  -0.036827   0.055687   5.667278 -13.843941
   3   0.000002   0.051446   0.000073  -7.353936
   4   0.027471   0.043993   0.423409  -6.207061
   5  -0.009357   0.040649 -12.736464   3.762324
   6  -0.000001   0.040278  -0.001621  -2.224536
   7   0.001277   0.039898  -1.266298  -3.916647
   8  -0.031609   0.020149  51.647411 -21.635780
   9  -0.000021   0.019859   0.035760 -61.239749
  10  -0.010397   0.019376  18.573156 -50.591165
  11  -0.026641   0.015808  -8.871055  -6.654795
-----------------------------------------------

zpv correction to isotropic shift :  -4.840215 ppm
-----------------------------------------------
```

So the absolute shielding constant of carbon in methanol needs to be corrected by -4.8 ppm due to zero-point vibration. From the mean and mean square displacements and the first and second derivatives of the shieldings with respect to the normal modes, one can also identify degrees of freedom which give rise to larger contributions of the vibrational correction.

A similar input for the HH spin-spin coupling constants would look like this :

```
!TPSS pcJ-2 Autoaux ExtremeSCF NMR

%geom inhess read
 inhessname "vpt2_methanol_FF.hess"
end

%maxcore 4096

%vpt2
  VPT2          on
  AvgProp       JCOUPLING
  AnharmDisp    0.05
  HessianCutoff 1e-12
end

%method
  Z_Tol         1e-12
end

%eprnmr
  Tol           1e-10
end

* xyz 0 1
C  -1.09849212248373      0.14540972773089     -0.00000275092982
O   0.32138758531316      0.08706714755687     -0.00001212477411
```

(continues on next page)

(continued from previous page)

```
H   0.66732439683790      0.98510769198508      0.00001819506998
H  -1.45583606337199     -0.88374271593276      0.00000595999622
H  -1.49206267729630      0.64725244577978      0.89143349761200
H  -1.49208273899904      0.64724452288014     -0.89144277697426
*

%eprnmr
Nuclei = all H {ssfc}
end
```

As mentioned above, the exact same procedure is also available for A-tensors. Here is an example for the NH_2 radical with the VPT2 input file `vpt2_NH2_FF.inp` :

```
!UKS BP86 def2-svp def2/J ExtremeSCF defgrid3

%vpt2
    VPT2                On
end

%method
    Z_Tol               1e-12
end

* xyz 0 2
N   -0.01498947828047     -0.01894387811818      0.00000000000000
H   1.03197835263254      0.00908678452370      0.00000000000000
H   -0.22855980523269      1.00639225931822      0.00000000000000
*
```

and the input file - something like `vpt2_NH2_A.inp` - for the level of theory that will be used in the A-tensor computation:

```
!UKS BP86 def2-SVP TightSCF

%geom inhess read
 inhessname "vpt2_NH2_FF.hess"
end

%vpt2
    VPT2                On
    AvgProp             Atensor
end

*xyz 0 2
N   -0.01498947828047     -0.01894387811818      0.00000000000000
H   1.03197835263254      0.00908678452370      0.00000000000000
H   -0.22855980523269      1.00639225931822      0.00000000000000
*

%eprnmr
Nuclei = all N { aiso, adip }
Nuclei = all H { aiso, adip }
end
```

and similarly for the g-tensor if `Atensor` is replaced by `Gtensor` in the `%vpt2` block (the `%eprnmr` block can be omitted then).

Note that a convenient way to compute vibrational corrections is the usage of a compound script. With an input file called `NH2.inp` :

```
* xyz 0 2
N   0.00312611577632      0.00395297373474      0.00000000000000
H   1.01930353842041      0.00049997276783      0.00000000000000
H   -0.23400058507735      0.99208221922117      0.00000000000000
*

%Compound "NH2.cmp"
```

and the corresponding compound script `NH2.cmp`:

```
New_Step
!UHF def2-SVP VeryTightSCF

%vpt2
    VPT2                On
end

%method
    Z_Tol               1e-12
end

* xyz 0 2
N   0.00312611577632      0.00395297373474      0.00000000000000
H   1.01930353842041      0.00049997276783      0.00000000000000
H   -0.23400058507735      0.99208221922117      0.00000000000000
*
Step_End

New_Step
!UHF def2-SVP VeryTightSCF

%geom inhess read
 inhessname "NH2_Compound_1.hess"
end

%vpt2
    VPT2                On
    AvgProp             Atensor
end

*xyz 0 2
N   0.00312611577632      0.00395297373474      0.00000000000000
H   1.01930353842041      0.00049997276783      0.00000000000000
H   -0.23400058507735      0.99208221922117      0.00000000000000
*

%eprnmr
Nuclei = all N { aiso, adip }
Nuclei = all H { aiso, adip }
end

Step_End

END
```

a similar result can be obtained in one calculation.

Note

Make sure the correct hessian file names are given and the input files MUST not contain a compound block. You can also rerun the VPT2 analysis using `orca_vpt2` directly. If you have an anharmonic force field cal-

culation named `myjob_ff` and a property derivative calculation named `myjob_prop` just call `orca_vpt myjob_ff.vpt2 myjob_prop.vpt2`.

5.20 Electrical Properties - Electric Moments and Polarizabilities

A few basic electric properties can be calculated in ORCA although this has never been a focal point of development. The properties can be accessed straightforwardly through the `%elprop` block:

```
! B3LYP DEF2-SVP TightSCF

%elprop Dipole        true # dipole moment
        Quadrupole    true # quadrupole moment

        Polar         true # dipole-dipole polarizability
                         1 # equivalent to true (for backward compatibility)
                           # Note: the flags "polar 2" and "polar 3" for␣
↪seminumeric
                           # and fully numeric polarizabilities are not supported
                           # anymore! For numerical polarizability calculations
                           # please use the respective compound scripts

        Hyperpol      true # the dip/dip/dip hyperpolarizability

        PolarVelocity true # polarizability w.r.t. velocity perturbations
        PolarDipQuad  true # dipole-quadrupole polarizability
        PolarQuadQuad true # quadrupole-quadrupole polarizability
        end
* int 0 1
   C  0  0  0  0       0         0
   H  1  0  0  1.09  109.4712    0
   H  1  2  0  1.09  109.4712    0
   H  1  2  3  1.09  109.4712  120
   H  1  2  3  1.09  109.4712  240
*
```

The polarizability (dipole-dipole, dipole-quadrupole, quadrupole-quadrupole) is calculated analytically through solution of the coupled-perturbed (CP-)SCF equations for HF and DFT runs (see *CP-SCF Options*) and through the CP-CASSCF equations for CASSCF runs (see *CASSCF Linear Response*). Analytic polarizabilities are also available for CCSD/CCSD(T) (see *AUTOCI Response Properties via Analytic Derivatives*), RI-MP2 and DLPNO-MP2, as well as double-hybrid DFT methods. For canonical MP2 one can use AUTOCI for analytic calculations (see *AUTOCI Response Properties via Analytic Derivatives*) or differentiate the analytical dipole moment calculated with relaxed densities. For other correlation methods only a fully numeric approach is possible.

```
---------------------------------------------------
STATIC POLARIZABILITY TENSOR (Dipole/Dipole)
---------------------------------------------------

Method              : SCF
Type of density     : Electron Density
Type of derivative  : Electric Field (Direction=X)
Multiplicity        :   1
Irrep               :   0
Relativity type     :
Basis               : AO

The raw cartesian tensor (atomic units):
      12.852429555        -0.002199911         0.000000170
      -0.002199911        12.860507003        -0.000000346
```

(continues on next page)

(continued from previous page)

```
       0.000000170        -0.000000346        12.868107945
diagonalized tensor:
      12.851869269        12.861067290        12.868107945

Orientation:
       0.969064588        -0.246807263         0.000017958
       0.246807263         0.969064586        -0.000050696
      -0.000004890         0.000053560         0.999999999

Isotropic polarizability :  12.86035
```

As expected the polarizability tensor is isotropic.

Dipole-quadrupole polarizability tensors are printed as a list of 18 different components, with the first index running over x,y,z and the second index running over xx,yy,zz,xy,xz,yz. This is known as the "pure Cartesian" version of the tensor, although other conventions may exist in the literature that differ from the ORCA values by a constant factor.

```
---------------------------------------------------
STATIC POLARIZABILITY TENSOR (Dipole/Quadrupole)
---------------------------------------------------

Method              : SCF
Type of density     : Electron Density
Type of derivative  : Electric Field (Direction=X)
Multiplicity        :   1
Irrep               :   0
Relativity type     :
Basis               : AO

The raw cartesian tensor (atomic units):
  X- X X :     11.577165985
  X- Y Y :     -5.795339382
  X- Z Z :     -5.797320742
  X- X Y :      0.001285565
  X- X Z :      0.000000155
  X- Y Z :     -0.000000077
  Y- X X :      0.001386387
  Y- Y Y :      8.200445841
  Y- Z Z :     -8.198375727
  Y- X Y :     -5.794687548
  Y- X Z :      0.000000228
  Y- Y Z :     -0.000000121
  Z- X X :     -0.000000151
  Z- Y Y :      0.000000627
  Z- Z Z :     -0.000000812
  Z- X Y :     -0.000000312
  Z- X Z :     -5.798359323
  Z- Y Z :     -8.205110537
```

After this, the "traceless" version of the tensor is printed, which is usually denoted by $A_{x,xx}, A_{x,xy}$ etc. in the literature[682, 683, 684]. This is the preferred format for reporting dipole-quadrupole polarizability tensors. Certain references use the opposite sign convention than reported here, but generally the conventions of traceless polarizability tensors are more unified than those of pure Cartesian polarizability tensors.

```
-------------------------------------------------------------
STATIC TRACELESS POLARIZABILITY TENSOR (Dipole/Quadrupole)
-------------------------------------------------------------

Method              : SCF
Type of density     : Electron Density
Type of derivative  : Electric Field (Direction=X)
```

(continues on next page)

(continued from previous page)

```
Multiplicity       :   1
Irrep              :   0
Relativity type    :
Basis              : AO

The raw cartesian tensor (atomic units):
  X- X X :     17.373496046
  X- Y Y :     -8.685262003
  X- Z Z :     -8.688234043
  X- X Y :      0.001928347
  X- X Z :      0.000000232
  X- Y Z :     -0.000000116
  Y- X X :      0.000351329
  Y- Y Y :     12.298940512
  Y- Z Z :    -12.299291841
  Y- X Y :     -8.692031322
  Y- X Z :      0.000000342
  Y- Y Z :     -0.000000181
  Z- X X :     -0.000000058
  Z- Y Y :      0.000001109
  Z- Z Z :     -0.000001050
  Z- X Y :     -0.000000468
  Z- X Z :     -8.697538984
  Z- Y Z :    -12.307665806
```

The quadrupole-quadrupole polarizability tensor is similarly printed in both the pure Cartesian and traceless forms. Again, the traceless form (usually denoted as $C_{xx,xx}$, $C_{xx,xy}$ etc.[682, 683, 684]) is the preferred format for reporting.

```
---------------------------------------------------
STATIC POLARIZABILITY TENSOR (Quadrupole/Quadrupole)
---------------------------------------------------

The order in each direction is XX, YY, ZZ, XY, XZ, YZ

Method             : SCF
Type of density    : Electron Density
Type of derivative : Quadrupolar Field (Direction=X)
Multiplicity       :   1
Irrep              :   0
Relativity type    :
Basis              : AO

The raw cartesian tensor (atomic units):
      60.656194448         8.024072323         8.017351959        -0.002591466    ␣
↪     0.000000801        -0.000000184
       8.024072323        55.906127614        12.837825709        -6.821368242    ␣
↪    -0.000000954        -0.000000529
       8.017351959        12.837825709        55.938851507         6.815300773    ␣
↪     0.000000232         0.000000422
      -0.002591466        -6.821368242         6.815300773        16.716647772    ␣
↪     0.000000169        -0.000000030
       0.000000801        -0.000000954         0.000000232         0.000000169    ␣
↪    16.715850196         6.818791255
      -0.000000184        -0.000000529         0.000000422        -0.000000030    ␣
↪     6.818791255        21.534628724
diagonalized tensor:
      11.893291534        13.566719080        26.357187387        46.234564137    ␣
↪    52.663246003        76.753292120

Orientation:
      -0.000000018        -0.000019986        -0.000000013        -0.001433194    ␣
```

(continues on next page)

(continued from previous page)

```
↪    0.817436692         0.576016666
      0.000000006         0.219691224         0.000000038         0.673107563    ␣
↪   -0.405967809         0.577799371
      0.000000008        -0.219450737        -0.000000021        -0.671194117    ␣
↪   -0.408640606         0.578232381
     -0.000000035         0.950566746        -0.000000034        -0.310519906    ␣
↪   -0.000497156        -0.000034103
      0.816443231         0.000000038         0.577425709        -0.000000037    ␣
↪    0.000000027         0.000000000
     -0.577425709        -0.000000003         0.816443231        -0.000000036    ␣
↪    0.000000002        -0.000000003

Isotropic polarizability :  37.91138



-------------------------------------------------------------
STATIC TRACELESS POLARIZABILITY TENSOR (Quadrupole/Quadrupole)
-------------------------------------------------------------

The order in each direction is XX, YY, ZZ, XY, XZ, YZ

Method              : SCF
Type of density     : Electron Density
Type of derivative  : Quadrupolar Field (Direction=X)
Multiplicity        :   1
Irrep               :   0
Relativity type     :
Basis               : AO

The raw cartesian tensor (atomic units):
      26.331642600       -13.160050722       -13.171591878         0.000221134    ␣
↪    0.000000581        -0.000000065
     -13.160050722        22.733889017        -9.573838294        -5.113861448    ␣
↪   -0.000000735        -0.000000324
     -13.171591878        -9.573838294        22.745430172         5.113640314    ␣
↪    0.000000154         0.000000389
       0.000221134        -5.113861448         5.113640314        12.537485829    ␣
↪    0.000000127        -0.000000022
       0.000000581        -0.000000735         0.000000154         0.000000127    ␣
↪   12.536887647         5.114093442
      -0.000000065        -0.000000324         0.000000389        -0.000000022    ␣
↪    5.114093442        16.150971543
```

Note

- **Like the quadrupole moments themselves, the dipole-quadrupole and quadrupole-quadrupole polarizabilities depend on the gauge origin of the `%elprop` module. The latter can be changed using the `Origin` keyword in `%elprop`; see section *Electrical Properties - Electric Moments and Polarizabilities*.**

At the SCF level, dynamic (frequency-dependent) dipole polarizabilities can be computed using either purely real or purely imaginary frequencies.

```
%elprop
 polar 1
 freq_r 0.08  # purely real frequencies
 #freq_i 0.08 # purely imaginary frequencies
end
```

In the example above, the dynamic dipole polarizability tensor for a single real frequency of 0.8 a.u. is computed. For every frequency, linear response equations must be solved for all component of the perturbation operator (3 Cartesian components of the electric dipole). Note that imaginary-frequency polarizabilities are computed with the same costs as real-frequency polarizabilities. By a simple contour integration they can be used to compute dispersion coefficients.

For methods that do not support analytic polarizabilities, one can calculate numeric dipole-dipole and quadrupole-quadrupole polarizabilities, either by finite differentiation of dipole/quadrupole moments with respect to a finite dipole/quadrupole electric field, or by double finite differentiation of the total energy with respect to a finite dipole/quadrupole electric field. The latter can be done using compound scripts (see *Compound*, *Compound Examples*).

At the MP2 level, polarizabilities can currently be calculated analytically using the RI (*RI-MP2 and Double-Hybrid DFT Response Properties*) or DLPNO (*Local MP2 Second Derivatives and Response Properties*) approximations or in all-electron canonical calculations, but for canonical MP2 with frozen core orbitals the dipole moment has to be differentiated numerically in order to obtain the polarizability tensor. In general in such cases, you should keep in mind that tight SCF convergence is necessary in order to not get too much numerical noise in the second derivative. Also, you should experiment with the finite field increment in the numerical differentiation process.

As an example, the following Compound job can be used to calculate the seminumeric polarizability at the MP2 level (replacing the now deprecated usage of `Polar 2`):

```
*xyz 0 1
  O   0.00000000000000      0.05591162058341      0.05591162058342
  H   0.00000000000000     -0.06629333722358      1.01038171664016
  H   0.00000000000000      1.01038171664017     -0.06629333722358
*

%Compound "semiNumericalPolarizability.cmp"
  with
    method      = "MP2";
    basis       = "aug-cc-pVDZ cc-pVDZ/C";
    restOfInput = "VeryTightSCF PModel NoFrozenCore";
  end
```

with the file `semiNumericalPolarizability.cmp` containing:

```
# ----------------------------------------------------------------------
# Authors: Dimitrios G. Liakos and Frank Neese
# Date   : March-May of 2024
#
# This is a compound script that calculates the
#  raw porarizability tensor semi-numerically
#  using the dipole moments.
#
# The idea is the following:
#
# 1. Calculate dipole moment  in the field free case
#
# 2. Loop over directions I=X,Y,Z
#    - put a small E-field in direction I+Delta
#    - Solve equations to get the dipole moment D+
#    - put s small E-field in direction I-Delta
#    - Solve equations to get the dipole moment D-
#    - Polarizability alpha(I,J). (D+(I)-D-(I))/(2Delta)
#    - Diagonalize to get eigenValues, eigenVectors, a_iso
#
# 3. Print polarisability
#
# NOTE: We use the last dipole_moment found in the file. If a specific
#       one is needed the 'myProperty' option should be accordingly
#       adjusted and remove the 'property_Base = true' option.
#
```

(continues on next page)

(continued from previous page)

```
# NOTE: This is not the most general version. It is adjusted for testsuite
#       with 'method' and 'mp2' blocks.
# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
# ----------------------    Variables    -----------------------------
#
# --- Variables to be adjusted (e.g. using 'with' ---------------------
Variable molecule    = "h2o.xyz";        # xyz file with coordinates
Variable charge      = 0;
Variable mult        = 1;
Variable method      = "HF";
Variable basis       = " " ;
Variable restOfInput = "NoFrozenCore VeryTightSCF";
Variable E_Field     = 0.0001;           # Size of Electric field
Variable myProperty  = "Dipole_Moment_Total";
Variable removeFiles = true;             # Remove files in the end of the␣
↪calculation
# ---------------------------------------------------------------------
# -------------- Rest of the variables -------------------------------
Variable D_Free, D_Minus, D_Plus, a[3][3];    #dipole moment and polarizability
Variable aEigenValues[3], aEigenVectors[3][3], a_iso;
Variable FFieldStringPlus, FFieldStringMinus;
Variable res = -1;

# ---------------------------------------------------------------------
# Field Free
# ---------------------------------------------------------------------
New_Step
  !&{method} &{basis} &{restOfInput}
  %Method
    z_tol 1e-8
  End
  %MP2
    Density Relaxed
  End
  #*xyzFile &{charge} &{mult} &{molecule}
Step_End
res = D_Free.readProperty(propertyName=myProperty, property_Base=true);

# -------------------------------------------------------------------
# Loop over the x, y, z directions and create the appropriate string
# -------------------------------------------------------------------
for direction from 0 to 2 Do
  #Create the appropriate direction oriented field string
  if (direction = 0) then          #( X direction)
    write2String(FFieldStringPlus,  " %lf, 0.0, 0.0", E_Field);
    write2String(FFieldStringMinus, "-%lf, 0.0, 0.0", E_Field);
  else if (direction = 1) then     #( Y direction)
    write2String(FFieldStringPlus,  " 0.0,  %lf, 0.0", E_Field);
    write2String(FFieldStringMinus, " 0.0, -%lf, 0.0", E_Field);
  else                             #( Z direction)
    write2String(FFieldStringPlus,  " 0.0,  0.0,  %lf", E_Field);
    write2String(FFieldStringMinus, " 0.0,  0.0, -%lf", E_Field);
  EndIf
  # ----------------------------------------
  # Perform the calculations.
  # First the plus (+) one
  # ----------------------------------------
  ReadMOs(1);
  New_Step
    !&{method} &{basis} &{restOfInput}
```

(continues on next page)

(continued from previous page)

```
    %SCF
      EField = &{FFieldStringPlus}
    End
    %Method
      z_tol 1e-8
    End
    %MP2
      Density Relaxed
    End
  Step_End
  res = D_Plus.readProperty(propertyName=myProperty, property_Base=true);

  # ----------------------------------------
  # And the minus (-) one
  # ----------------------------------------
  ReadMOs(1);
  New_Step
    !&{method} &{basis} &{restOfInput}
    %SCF
      EField = &{FFieldStringMinus}
    End
    %Method
      z_tol 1e-8
    End
    %MP2
      Density Relaxed
    End
  Step_End
  res = D_Minus.readProperty(propertyName=myProperty, property_Base=true);

  # -------------------------------------------
  # Construct and store SCF polarizability
  # -------------------------------------------
  a[direction][0] = (D_Plus[0]-D_Minus[0])/(2*E_Field);
  a[direction][1] = (D_Plus[1]-D_Minus[1])/(2*E_Field);
  a[direction][2] = (D_Plus[2]-D_Minus[2])/(2*E_Field);

EndFor
# ----------------------------------------
# Diagonalize
# ----------------------------------------
a.Diagonalize(aEigenValues, aEigenVectors);

# ----------------------------------------
# Do some printing
# ----------------------------------------
print( "\n\n");
print( " -------------------------------------------------------\n");
print( "                   COMPOUND                            \n");
print( " Semi analytical calculation of polarizability\n");
print( " -------------------------------------------------------\n");
print( " Method:  %s\n", method);
print( " Basis :  %s\n", basis);
print( " The electric field perturbation used was:    %.5lf a.u.\n", E_Field);
print( " \n\n");

print( " -------------------------------------------------------\n");
print( " Raw electric semi-analytical polarizability tensor\n");
print( " -------------------------------------------------------\n");
For i from 0 to 2 Do
  print("%13.8lf  %13.8lf  %13.8lf\n", a[i][0], a[i][1], a[i][2]);
```

(continues on next page)

(continued from previous page)

```
EndFor
print( " -------------------------------------------------------\n");
print("\n");

print( " -------------------------------------------------------\n");
print( " Raw electric semi-analytical polarizability Eigenvalues\n");
print( " -------------------------------------------------------\n");
print("%13.8lf  %13.8lf  %13.8lf\n", aEigenValues[0], aEigenValues[1],␣
↪aEigenValues[2]);
print( " -------------------------------------------------------\n");
print("\n");

print( " -------------------------------------------------------\n");
print( " Raw electric semi-analytical polarizability Eigenvectors\n");
print( " -------------------------------------------------------\n");
For i from 0 to 2 Do
  print("%13.8lf  %13.8lf  %13.8lf\n", aEigenVectors[i][0], aEigenVectors[i][1],␣
↪aEigenVectors[i][2]);
EndFor

print( "\n a isotropic value : %.5lf\n",␣
↪(aEigenValues[0]+aEigenValues[1]+aEigenValues[2])/3.0);
#
#
# ---------------------------------------------------
#  Maybe remove unneccesary files
# ---------------------------------------------------
if (removeFiles) then
  sys_cmd("rm *_Compound_* *.bas* ");
EndIf

End
```

For more details on Compound jobs in general, see *Compound*.

For other correlation methods, where not even relaxed densities are available, only a fully numeric approach (using compounds scripts) is possible and requires obnoxiously tight convergence.

Note that polarizability calculations have higher demands on basis sets. A rather nice basis set for this property is the Sadlej one (see *Orbital Basis Sets*). In relation to electric properties, it might be interesting to know that it is possible to carry out finite electric field calculations in ORCA. See section *Finite Electric Fields* for more information on such calculations.

The static hyperpolarizability tensor is calculated as an analytical third derivative of the energy with respect to three dipole perturbations. It is currently available for SCF methods (HF, DFT). For example, consider the input:

```
! RHF def2-SVP TightSCF

%elprop
  hyperpol 1
end

*xyz 0 1
  C       0.000000     -0.000000     -0.528012
  O      -0.000000      0.000000      0.650528
  H      -0.000000      0.931690     -1.116318
  H      -0.000000     -0.931690     -1.116318
*
```

The hyperpolarizability can be found printed in the output as

```
---------------------------------------------------
         STATIC HYPERPOLARIZABILITY TENSOR
---------------------------------------------------

Method              : SCF
Type of density     : Electron Density
Type of derivative  : Electric Field (Direction=X)
Multiplicity        :   1
Irrep               :   0
Basis               : AO

  The raw Cartesian tensor (atomic units):

     ( x x x ):          -0.000000
     ( x x y ):          -0.000000
     ( x x z ):           7.076938
     ( x y x ):          -0.000000
     ( x y y ):          -0.000000
     ( x y z ):          -0.000000
     ( x z x ):           7.076938
     ( x z y ):          -0.000000
     ( x z z ):          -0.000000
     ( y x x ):          -0.000000
     ( y x y ):          -0.000000
     ( y x z ):          -0.000000
     ( y y x ):          -0.000000
     ( y y y ):           0.000000
     ( y y z ):          49.371503
     ( y z x ):          -0.000000
     ( y z y ):          49.371503
     ( y z z ):           0.000000
     ( z x x ):           7.076938
     ( z x y ):          -0.000000
     ( z x z ):          -0.000000
     ( z y x ):          -0.000000
     ( z y y ):          49.371503
     ( z y z ):           0.000000
     ( z z x ):          -0.000000
     ( z z y ):           0.000000
     ( z z z ):          44.368713
```

5.20.1 Atomic Dipole Moment and Polarizabilities

ORCA includes a mechanism by which electric properties can be decomposed into atomic contributions. In a nutshell, the method relies on the partitioning of real space into atomic regions. These regions are defined by a smooth weighting function for atom A

$$f_A(\mathbf{r}) = \frac{\varrho_A^{(0)}(\mathbf{r})}{\sum_B \varrho_B^{(0)}(\mathbf{r})}$$

The quantities $\varrho_A^{(0)}(r)$ are spherically symmetric atomic densities (expanded in a reasonably large set of Gaussian s-functions) for the neutral atoms that exit pretabulated in the code.

Using these partitioning functions, an atomic multipole moment is defined as:

$$Q_l^m(A) = \sum_{\mu\nu} P_{\mu\nu} \int \mu(\mathbf{r}) \hat{q}_l^m \nu(\mathbf{r}) f_A(\mathbf{r})\, d\mathbf{r}$$

Here **P** is the density matrix and $\hat{q}_l^m$ the operator for a multipole moment of order l with component m. For example, for the dipole moment l=1 and m=0,+1,-1. Likewise, for a quadrupole moment l=2, m=2,1,0,-1,-2 and so on. The

partitioned integration over space is carried out by numerical integration. In the limit of exact numerical integration, the construction guarantees that the atomic moments sum to the overall multipole moment. Because numerical integration introduces a small error, the accumulated moments

$$Q_l^m = \sum_A Q_l^m(A)$$

Will show small deviation from the analytically integrated moments that the program also calculates prints.

Clearly, the same strategy can be used for response properties. For example, for polarizabilities, one decomposes it into the atomic components as follows:

$$\alpha_{KL}(A) = -\sum_{\mu\nu} \frac{\partial P_{\mu\nu}}{\partial F_K} \int \mu(\mathbf{r})\mathbf{r}_L \nu(\mathbf{r}) f_A(\mathbf{r})\, d\mathbf{r}$$

Here **F** is an external electric field and the derivative of the density matrix is the response density with respect to this field.

In order to trigger this calculation use:

```
%elprop
Polar true
PolarAtom true
Dipole true
DipoleAtom true
Quadrupole true
QuadrupoleAtom true
end
```

Note

- Atomic electric properties are available presently at the SCF level for dipole moments, quadrupole moments and polarizabilities
- The dipole moment is well-known to be gauge dependent for charged molecules. Hence, the way the dipole moment is decomposed here, the atomic dipole moments refer to the global origin and there is a gauge dependence for charged atoms. Consequently, one needs to proceed with some caution in the interpretation of these local dipole moments.

PAPER:

- Grigorash, D.; Müller, S.; Heid, E.; Neese, F.; Liakos, D.; Riplinger, C.; Garcia-Rates, M.; Paricaud, P. Erling H.S.; Smirnova, I.; Yan, W. A comprehensive approach to incorporating intermolecular dispersion into the openCOSMO-RS model. Part 2: Atomic polarizabilities, 2025, submitted

5.20.2 Elprop Keywords

Calculation of first order (electric dipole and quadrupole moments) and second order (polarizabilities) electric properties can be requested in the `%elprop` input block. The second order properties can be calculated through the solution of the CP-SCF (see *CP-SCF Options*) or CP-CASSCF (see *CASSCF Linear Response*) equations. Details are shown below:

```
%elprop
  Dipole     true
  Quadrupole true
  Polar      true
  PolarVelocity true # polarizability w.r.t. velocity perturbations
  PolarDipQuad  true # dipole-quadrupole polarizability
  PolarQuadQuad true # quadrupole-quadrupole polarizability
```

(continues on next page)

(continued from previous page)

```
  Hyperpol      true # the dip/dip/dip hyperpolarizability
  freq_r 0.00   # purely real frequency (default: static calculation)
  freq_i 0.00   # purely imaginary frequency (default: static calculation)
  Solver  CG    # CG(conjugate gradient)
                # other options: DIIS or POPLE(default)
  MaxDIIS  5    # max. dimension of DIIS method
  Shift   0.2   # level shift used in DIIS solver
  Tol     1e-3  # Convergence of the CP-SCF equations
                # (norm of the residual)
  MaxIter 64    # max. number of iterations in CPSCF
  PrintLevel 2
  Origin  CenterOfElCharge  # center of electronic charge
          CenterOfNucCharge # center of nuclear charge
          CenterOfSpinDens        # center of spin density
          CenterOfMass      # center of mass (default)
          N                 # position of atom N (starting at 0)
          X,Y,Z             # explicit position of the origin
                            # in coordinate input units (Angstrom by default)
  end
```

The most efficient and accurate way to calculate the polarizability analytically is to use the coupled-perturbed SCF method. The most time consuming and least accurate way is the numerical second derivative of the total energy. Note that the numerical differentiation requires: (a) tightly or even very tightly converged SCF calculations and (b) carefully chosen field increments. If the field increment is too large then the truncation error will be large and the values will be unreliable. On the other hand, if the field increment is too small the numerical error associated with the finite difference differentiation will get unacceptably large up to the point where the whole calculation becomes useless.

5.21 Nuclear Magnetic Resonance (NMR) Parameters

5.21.1 NMR Chemical Shifts

NMR chemical shifts at the HF, DFT (standard GGA and hybrid functionals), CASSCF, as well as the RI- and DLPNO-MP2 and double-hybrid DFT levels (see section *MP2 Level Magnetic Properties* and references therein), plus Coupled Cluster (GIAO-CCSD, GIAO-CCSD(T), see *AUTOCI Response Properties via Analytic Derivatives*) can be obtained from the EPR/NMR module of ORCA. For the calculation of the NMR shielding tensor the program utilizes Gauge Including Atomic Orbitals (GIAOs - sometimes also referred to as London orbitals). [685, 686, 687] In this approach, field dependent basis functions are introduced, which minimizes the gauge origin dependence and ensures rapid convergence of the results with the one electron basis set. [688] Note that GIAOs are **NOT** currently available with CASSCF linear response and a gauge origin must be provided in the `%eprnmr` block (see *CASSCF Linear Response*). GIAOs for CASSCF response are coming soon to ORCA! A complete list of possible keywords for the eprnmr module can be found in *EPRNMR - keywords for magnetic properties*.

The use of the chemical shift module is simple:

```
# Ethanol - Calculation of the NMR chemical shieldings at the HF/SVP level of␣
↪theory
! RHF SVP Bohrs NMR

* xyz 0 1
C        -1.22692181     0.24709455    -0.00000000
C        -0.01354839    -0.54677253     0.00000000
H        -2.09280406    -0.41333631     0.00000000
H        -1.24962478     0.87541936    -0.88916500
H        -1.24962478     0.87541936     0.88916500
O         1.09961824     0.30226226    -0.00000000
H         0.00915178    -1.17509696     0.88916500
```

(continues on next page)

(continued from previous page)

```
H         0.00915178    -1.17509696    -0.88916500
H         1.89207683    -0.21621566     0.00000000
*
```

The output for the shieldings contains detailed information about the para- and diamagnetic contribution, the orientation of the tensor, the eigenvalues, its isotropic part etc. For each atom, an output block with this information is given :

```
--------------
 Nucleus   0C :
 --------------

Diamagnetic contribution to the shielding tensor (ppm) :
           209.647        -10.519         0.000
           -26.601        215.858         0.000
            -0.000          0.000       200.382

Paramagnetic contribution to the shielding tensor (ppm):
            59.273         18.302        -0.000
            13.380          6.063        -0.000
             0.000         -0.000        -2.770

Total shielding tensor (ppm):
           268.920          7.783        -0.000
           -13.220        221.921        -0.000
             0.000          0.000       197.611


 Diagonalized sT*s matrix:

 sDSO           200.382          214.507          210.998  iso=     208.629
 sPSO            -2.770            7.279           58.057  iso=      20.855
        ---------------  ---------------  ---------------
 Total          197.611          221.786          269.055  iso=     229.484
```

Note that all units are given in ppm and the chemical shieldings given are *absolute* shieldings (see below). At the end of the atom blocks, a summary is given with the isotropic shieldings and the anisotropy [689] for each nucleus:

```
Nucleus  Element    Isotropic     Anisotropy
  -------  -------  ------------   ------------
      0       C          229.484         59.356
      1       C          227.642         62.878
      2       H           56.015         12.469
      3       H           55.460         15.284
      4       H           55.460         15.284
      5       O          334.125        110.616
      6       H           47.337         27.101
      7       H           47.337         27.101
      8       H           64.252         32.114
```

Thus, the absolute, isotropic shielding for the ^{13}C nuclei are predicted to be 229.5 and 227.6 ppm and for ^{17}O it is 334.1 ppm. While basis set convergence using GIAOs is rapid and smooth, it is still recommended to do NMR calculations with basis sets including tight exponents, such as the purpose-built pcSseg-n. However, TZVPP or QZVP should be sufficient in most cases. [690, 691]

An important thing to note is that in order to compare to experiment, a standard molecule for the type of nucleus of interest has to be chosen. In experiment, NMR chemical shifts are usually determined relative to a standard, for example either CH_4 or TMS for proton shifts. Hence, the shieldings for the molecule of interest and a given standard molecule are calculated, and the relative shieldigs are obtained by subtraction of the reference value from the computed value. It is of course important that the reference and target calculations have been done with the same basis set and functional. This also helps to benefit from error cancellation if the standard is chosen appropriately (one

option is even to consider an "internal standard" - that is to use for example the ^{13}C shielding of a methyl group inside the compound of interest as reference).

Let us consider an example - propionic acid (CH_3-CH_2COOH). In databases like the AIST (http://sdbs.db.aist.go.jp) the ^{13}C spectrum in $CDCl_3$ can be found. The chemical shifts are given as δ_1 = 8.9 ppm, δ_2 = 27.6 ppm, δ_3 = 181.5 ppm. While intuition already tells us that the carbon of the carboxylic acid group should be shielded the least and hence shifted to lower fields (larger δ values), let's look at what calculations at the HF, BP86 and B3LYP level of theory using the SVP and the TZVPP basis sets yield:

method	σ_1	σ_2	σ_3
HF/SVP	191.7	176.6	23.7
HF/TZVPP	183.5	167.1	9.7
B86/SVP	181.9	165.8	26.5
B86/TZVPP	174.7	155.5	7.6
B3LYP/SVP	181.8	165.8	22.9
B3LYP/TZVPP	173.9	155.0	2.9

Looking at these results, we can observe several things - first of all, the dramatic effect of using too small basis sets, which yields differences of more than 10 ppm. Second, the results obviously change a lot upon inclusion of electron correlation by DFT and are functional dependent. Last but not least, these values have nothing in common with the experimental ones (they change in the wrong order), as the calculation yields *absolute shieldings* like in the table above, but the experimental ones are *relative shifts*, in this case relative to TMS.

In order to obtain the relative shifts, we calculate the shieldings σ_{TMS} of the standard molecule (TMS HF/TZVPP: 194.1 ppm, BP86/TZVPP: 184.8 ppm, B3LYP/TZVPP: 184.3 ppm) and by using $\delta_{mol} = \sigma_{ref} - \sigma_{mol}$ we can evaluate the chemical shifts (in ppm) and directly compare to experiment:

method	δ_1	δ_2	δ_3
HF/TZVPP	10.6	27.0	184.4
B86/TZVPP	10.1	29.3	177.2
B3LYP/TZVPP	10.4	29.3	181.4
Exp.	8.9	27.6	181.5

A few notes on the GIAO implementation in ORCA are in order. The use of GIAOs lead to some fairly complex molecular one- and two-electron integrals and a number of extra terms on the right hand side of the coupled-perturbed SCF equations. The two-electron contributions in particular can be time consuming to calculate. Thus, the RIJK, RIJDX, and RIJCOSX approximations were implemented and tested.[692] By default, the approximation used for the SCF is also applied to the GIAO integrals, but the latter can be changed using the `GIAO_2el` keyword in the eprnmr input block (see section *EPRNMR - keywords for magnetic properties*). Note that, while the default COSX grids are typically sufficiently accurate for chemical shifts, the use of `defgrid3` is recommended for special cases or post-HF calculations.

The user can finely control for which nuclei the shifts are calculated (although this will not reduce the computational cost very much, which is dominated by the CP-SCF equations for the magnetic field). This works in exactly the same way as for the hyperfine and quadrupole couplings described in the next section. For example:

```
! B3LYP def2-TZVP TightSCF

* int 0 1
C  0 0 0  0 0 0
C  1 0 0  1.35 0 0
H  1 2 0  1.1 120 0
H  1 2 3  1.1 120 180
H  2 1 3  1.1 120 0
H  2 1 3  1.1 120 180
*
```

(continues on next page)

(continued from previous page)

```
%eprnmr
  Ori    = GIAO
  Nuclei = all C { shift }
  Nuclei = all H { shift }
  end
```

NMR chemical shifts are also implemented in combination with implicit solvent models, hence the NMR keyword can be combined with the CPCM input block. Note that for calculations including implicit solvent, it is highly recommended to also optimize the geometries using the same model. Regardless, explicit solvent–solute interactions observable in NMR (e.g. H-bonds), cannot be modelled with such a model: solvent molecules must be included explicitly in the calculation.

5.21.2 NMR Spin-Spin Coupling Constants

The indirect spin-spin coupling constants observed in NMR spectra of molecules in solution consist of four contributions: The diamagnetic spin orbit term:

$$\hat{H}_{DSO} = \frac{1}{2} \sum_{ikl} \frac{(\vec{M}_k \times \vec{r}_{ik})(\vec{M}_l \times \vec{r}_{il})}{r_{ik}^3 \, r_{il}^3} \tag{5.91}$$

The paramagnetic spin orbit term:

$$\hat{H}_{PSO} = \sum_{ik} \frac{\vec{M}_k \, \vec{l}_{ik}}{r_{ik}^3} \tag{5.92}$$

The Fermi contact term:

$$\hat{H}_{FC} = \frac{8\,\pi}{3} \sum_{ik} \delta(r_i - r_k) \, \mathbf{m}_k \tag{5.93}$$

And the spin dipole term:

$$\hat{H}_{SD} = \sum_{ik} \mathbf{m}_k^T \, \frac{3\,\mathbf{r}_{ik}\,\mathbf{r}_{ik}^T - r_{ik}^2}{r_{ik}^5} \, \mathbf{s}_i \tag{5.94}$$

While the Fermi contact term is usually the most significant, all contributions can be computed at the HF and DFT level of theory using ORCA. For this purpose, the keyword `ssall` has to be invoked in the eprnmr input block, while each of the four terms can be requested using `ssdso`, `sspso`, `ssfc`, and `sssd`, respectively. For example:

```
! PBE0 pcJ-1

*xyz 0 1
O 0.00000        0.00000        0.11779
H 0.00000        0.75545       -0.47116
H 0.00000       -0.75545       -0.47116
*

%eprnmr
 Nuclei = all O { ssall }
 Nuclei = all H { ssfc, sssd }
end
```

Results will be given in Hz. Note that the default isotopes used might not be the ones desired for the calculation of NMR properties, so it is recommended to define the corresponding isotopes using the `ist` flag. It is possible to also print the reduced coupling constants K (in units of $10^{19} \cdot \mathrm{T} \cdot \mathrm{J}^{-2}$), which are independent of the nuclear isotopes, using the flag `PrintReducedCoupling=True`.

The CP-SCF equations must be solved for one of the nuclei in each pair and are the bottleneck of the computation. Therefore, spin-spin coupling constants are calculated only between nuclei within a certain distance of eachother (5 Ångstrom by default). The latter can be changed using the `SpinSpinRThresh` keyword.

If mulitple nuclides are requested, it is also possible to select only certain element pairs (e.g. only C–H and H–H, without C–C) using the `SpinSpinElemPairs` keyword. Analogously, the `SpinSpinAtomPairs` keyword selects the actual pairs of nuclei to consider. The union of the latter two options is used to *reduce* the selection made using the `Nuclei` input, after which `SpinSpinRThresh` is applied.

Here is another example illustrating these additional options:

```
! B3LYP EPR-II

* xyz 0 1
C     -1.226922    0.247095   -0.000000
C     -0.013548   -0.546773    0.000000
H     -2.092804   -0.413336    0.000000
H     -1.249625    0.875419   -0.889165
H     -1.249625    0.875419    0.889165
O      1.099618    0.302262   -0.000000
H      0.009152   -1.175097    0.889165
H      0.009152   -1.175097   -0.889165
H      1.892077   -0.216216    0.000000
*

%eprnmr
  nuclei = all C { ssall, ist = 13 };
  nuclei = all H { ssall, ist = 1  };
  nuclei = all O { ssall, ist = 17 };
  PrintReducedCoupling true
  SpinSpinRThresh 3.0 # Angstrom
  SpinSpinElemPairs {C C} {O *} # "*" means any element
  SpinSpinAtomPairs {8 *}       # indices start from 0

  # Final selection:
  # C 0 - C 1
  # C 0 - O 5
  # C 1 - O 5
  # C 1 - H 8
  # H 3 - O 5
  # H 4 - O 5
  # O 5 - H 6
  # O 5 - H 7
  # O 5 - H 8
  # H 6 - H 8
  # H 7 - H 8
end
```

5.21.3 Simulating NMR Spectra

From the computed NMR shieldings and spin-spin coupling constants, the coupled NMR spectra can be simulated. The most typical NMR experiments are decoupled ^{13}C and coupled ^{1}H spectra, so we will focus on these here. For simulating a full NMR spectrum, we will use ethyl crotonate as an example, and three steps are required: 1) computation of the spin-spin coupling constants, 2) computation of the chemcial shieldings and 3) simulation of the spectrum using a spin-Hamiltonian formalism. Note that these steps can be carried out independently and different levels of theory can be used for shieldings and couplings and the order of steps 1 and 2 doesn't matter.

Furthermore, if the spectra are to be simulated with TMS as reference, the shieldings for TMS are required (the hydrogen and carbon values in this case are 31.77 and 188.10 ppm, respectively). Here is the input for the calculation for the coupling constants, which is named `ethylcrotonate_sscc.inp`:

```
! PBE pcJ-3 autoaux tightscf

*xyzfile 0 1 ethylcrotonate.xyz
```

(continues on next page)

(continued from previous page)

```
%eprnmr
 Nuclei = all H {ssall}
end
```

The spin-spin coupling constants will be stored in the file `ethylcrotonate_sscc.property.txt`, which the simulation of the NMR spectrum will need to read. The NMR shieldings and will be computed in the next step, for which the input `ethylcrotonate_nmr.inp` looks like this:

```
!TPSS pcSseg-3 autoaux tightscf NMR

*xyzfile 0 1 ethylcrotonate.xyz
```

The final NMR spectrum simulation is carried out using `orca_nmrspectrum`, which requires a separate input file `ethylcrotonate.nmrspec` which looks like this (note the required final END statement):

```
NMRShieldingFile = "ethylcrotonate_nmr"   #property file for shieldings
NMRCouplingFile  = "ethylcrotonate_sscc"  #property file for couplings
NMRSpecFreq = 80.00                       #spectrometer freq [MHz] (default 400)
PrintLevel = 0                            #PrintLevel for debugging info
NMRCoal = 1.0                             #threshold for merged lines [Hz]␣
↪(default 1)
NMRREF[1] 31.77                           #shielding for 1H reference [ppm]
NMRREF[6] 188.10                          #shielding for 13C reference [ppm]
NMREquiv                                  #lists of NMR-equivalent nuclei
1 {13 14 15} end                          #H 13,14,15 are equivalent (methyl)
2 {16 17} end                             #H 16 and 17 equivalent (ethyl)
3 {8 10 11} end                           #H 8,10,11 again equivalent methyl
end                                       #end equiv nucl block
END                                       #essential end of input
```

and contains the following keywords:
`NMRShieldingFile` and `NMRCouplingFile` denote the `.property.txt` files from which the shielding tensors and coupling constants will be read by the NMR spectrum module. If this line is not given, the program will exepect the shieldings or couplings in the property file of the current calculation.

`NMRSpecFreq` The NMR spectrometer frequency in MHz is decisive for the looks of the spectrum as shieldings are given in ppm and couplings are given in Hz. Default is 400 MHz.

`NMRCoal` If two lines are closer than this threshold (given in Hz) then the module will coalesce the lines to one line with double intensity. Default it 1 Hz.

`NMRREF[X]` Reference value for the absolute shielding of element X used in the relative shifts of the simulated spectrum. Typically, these are the absolute shielding values from a separate calculation of TMS, for example, and will be zero ppm in the simulated spectrum.

`NMREquiv` The user has to specify groups of NMR equivalent nuclei. These are typically atoms which interchange on the NMR timescale, like methyl group protons. The shieldings and couplings will be averaged for each group specified by the user.

with this input, `orca_nmrspectrum` is called with two arguments, the first one is a gbw file which contains all informations about the molecule, typically this is the gbw file of the nmr or the sscc calculation, and the name of the input file given above:

```
orca_nmrspectrum ethylcrotonate_nmr.gbw ethylcrotonate.nmrspec > output
```

If this calculation is carried out, the NMR spectrum module will read the files `ethylcrotonate_sscc.property.txt` and `ethylcrotonate_nmr.property.txt`, extract the shieldings and couplings, average the equivalent nuclei and set up an effective NMR spin Hamiltonian for each nucleus:

$$H_{NMR}(p,q) = \sigma_p \delta_{pq} + J_{pq} I_p I_q. \tag{5.95}$$

Caution

This includes all nuclei this nuclear spin couples to and should not contain too many spins (see `SpinSpinRThres`), as the spin Hamiltonian for each atom and the nuclei it couples to is diagonalized brute force. Afterwards, all spin excitations are accumulated and merged into the final spectrum for each element. For ethyl crotonate the NMR spectrum output looks like this:

```
-----------------------------------------------------
 NMR Peaks for atom type 1, ref value  31.7700 ppm :
-----------------------------------------------------
Atom   shift[ppm]   rel.intensity
   8        2.34    9.00
   8        2.36    9.00
   8        2.25    9.00
   8        2.27    9.00
   9        6.34    1.00
   9        6.36    3.00
   9        6.38    3.00
   9        6.41    1.00
   9        6.14    1.00
   9        6.16    3.00
   9        6.19    3.00
   9        6.21    1.00
  12        7.95    1.00
  12        7.85    3.00
  12        7.75    4.00
  12        7.65    4.00
  12        7.56    3.00
  12        7.47    1.00
  13        1.71    9.00
  13        1.61   18.00
  13        1.52    9.00
  16        4.56    4.00
  16        4.46   12.00
  16        4.37   12.00
  16        4.27    4.00

-----------------------------------------------------
 NMR Peaks for atom type 6, ref value 188.1000 ppm :
-----------------------------------------------------
Atom   shift[ppm]   rel.intensity
   2       25.70    1.00
   3      155.15    1.00
   4       19.96    1.00
   5       68.91    1.00
   6      174.39    1.00
   7      130.29    1.00

-----------------------------------------------------
 NMR Peaks for atom type 8, ref value 104.8826 ppm :
-----------------------------------------------------
Atom   shift[ppm]   rel.intensity
   0        0.00    5.00
   1      149.74    5.00
```

The first column denotes the atom number of the nucleus the signal arises from, the second column denotes the line position in ppm and the third line denotes the relative intensity of the signal. For oxygen, no reference value was given, so the program will autmatically set the lowest value obtained in the calculation as reference value.

Using gnuplot, for example, to plot the spectrum, the following plots for ^{13}C and ^{1}H are obtained[1] :

[1] The basic plot options for using gnuplot are `plot "mydata" using 2:3 w i, "mydata" using 2:3:1 with labels`

Fig. 5.62: Simulated $^{1}3C$ (top) and 1H (bottom) NMR spectra. Note that as only HH couplings have been computed, the spectra do not include any CH couplings and the carbon spectrum is also uncoupled.

This makes comparison to experiment and assessment of the computed parameters much easier, however, it is not as advanced as other codes and does not, for example, take conformational degrees of freedom etc. into account. Note that the corresponding property files can also be modified to tinker with the computed shieldings and couplings.

5.21.4 Visualizing shielding tensors using `orca_plot`

For the visualization it is recommended to perform an ORCA NMR calculation such that the corresponding `gbw` and `density` files required by `orca_plot` are generted by using the `!keepdens` keyword along with `!NMR`. If `orca_plot` is called in the interactive mode by specifying `orca_plot myjob.gbw -i` (note that `myjob.gbw`, `myjob.densities` and `myjob.property.txt` have to be in this directory), then following `1 - type of plot` and choosing `17 - shielding tensor`, confirming the name of the property file and then choosing `11 - Generate the plot` will generate a `.cube` file with shielding tensors depicted as ellipsoids at the corrsponding nuclei. These can be plotted for example using Avogadro, isosurface values of around 1.0 and somewhat denser grids for the cube file (like 100x100x100) are recommended. A typical plot for CF_3SCH_3 generated with Avogadro looks like this[2]:

[2] the same scheme can be applied to visualize polarisability tensors in the molecular framework using `orca_plot`.

Fig. 5.63: The shielding tensors of each atom in CF_3SCH_3 have been plotted as ellipsoids (a,b and c axis equivalent to the normalized principle axes of the shielding tensors) at the given nuclei.

5.21.5 Nucleus-independent chemical shielding

Aromaticity is a fundamental concept in chemistry and much attention has been payed to its analysis in quantum chemistry. One possibility to gain insight into aromaticity is to sample the ring current effect in NMR close to the π-system. As this is not done by inspecting the chemical shielding of any of the atoms, this quantity is called nucleus independent chemical shielding (NICS). Usually, a "dummy" atom is placed in the center of the ring and/or at some distance away from it. In ORCA, one needs to use a ghost atom, e.g. "`H:`" to ensure that the program generates DFT or COSX grid points in the region on interest. For technical reasons, this atom must also have at least one basis function, which can be set with `NewGTO`. An s-function with a sufficiently large exponent will not overlap with any other basis function in the molecule and will thus have no effect on the results, but only satisfy the technical requirement (note that the extra grid points may change some of the calculation results by increasing the accuracy of the numerical integration). If RI is used, a dummy fitting function must also be added to the AuxJ, AuxJK, and/or AuxC basis set. A typical input for benzene looks like this:

```
! TightSCF NMR PBE def2-TZVP RI def2/J
* gzmt 0 1
    H:                                NewGTO S 1 1 1e6 1 end NewAuxJGTO S 1 1 2e6 1␣
↪end
    H:  1   1                         NewGTO S 1 1 1e6 1 end NewAuxJGTO S 1 1 2e6 1␣
↪end
    C   1   1.39    2   90
    C   1   1.39    2   90  3   60
    C   1   1.39    2   90  4   60
    C   1   1.39    2   90  5   60
    C   1   1.39    2   90  3  -60
    C   1   1.39    2   90  6   60
    H   3   1.09    4   120 1   180
    H   4   1.09    3   120 1   180
    H   5   1.09    4   120 1   180
    H   6   1.09    5   120 1   180
    H   7   1.09    3   120 1   180
    H   8   1.09    7   120 1   180
*
```

5.21.6 Shielding tensor orbital decomposition

It is possible to decompose the NMR shielding tensor into orbital- or orbital pair contributions. One such option is the Natural Chemical Shielding analysis (see Section *Natural Chemical Shielding Analysis (NCS)*), while another is presented here. The shielding tensor for nucleus A can be exactly decomposed as follows:

$$\begin{aligned}
\boldsymbol{\sigma}^A &= \sum_p \left(\boldsymbol{\sigma}_p^{A,\text{para}} + \boldsymbol{\sigma}_p^{A,\text{dia}}\right) \\
\boldsymbol{\sigma}_i^{A,\text{para/dia}} &= \boldsymbol{\sigma}_{ii}^{A,\text{para/dia}} + \frac{1}{2}\sum_{j\neq i}\left(\boldsymbol{\sigma}_{ij}^{A,\text{para/dia}} + \boldsymbol{\sigma}_{ji}^{A,\text{para/dia}}\right) + \sum_a \left(\boldsymbol{\sigma}_{ia}^{A,\text{para/dia}} + \boldsymbol{\sigma}_{ai}^{A,\text{para/dia}}\right) \\
\boldsymbol{\sigma}_a^{A,\text{para/dia}} &= \boldsymbol{\sigma}_{aa}^{A,\text{para/dia}} + \frac{1}{2}\sum_{b\neq a}\left(\boldsymbol{\sigma}_{ab}^{A,\text{para/dia}} + \boldsymbol{\sigma}_{ba}^{A,\text{para/dia}}\right) \\
\boldsymbol{\sigma}_{pq}^{A,\text{para}} &= D_{pq}^{\mathbf{B}} h_{pq}^{\mathbf{M}_A} \\
\boldsymbol{\sigma}_{pq}^{A,\text{dia}} &= D_{pq} h_{pq}^{\mathbf{B},\mathbf{M}_A} \\
D_{pq} &= \sum_{\mu\nu\kappa\lambda} c_{\mu p} S_{\mu\nu} D_{\nu\kappa} S_{\kappa\lambda} c_{\lambda q} \\
D_{pq}^{\mathbf{B}} &= \sum_{\mu\nu\kappa\lambda} c_{\mu p} S_{\mu\nu} D_{\nu\kappa}^{\mathbf{B}} S_{\kappa\lambda} c_{\lambda q} \\
h_{pq}^{\mathbf{M}_A} &= \sum_{\mu\nu} c_{\mu p} h_{\mu\nu}^{\mathbf{M}_A} c_{\nu q} \\
h_{pq}^{\mathbf{B},\mathbf{M}_A} &= \sum_{\mu\nu} c_{\mu p} h_{\mu\nu}^{\mathbf{B},\mathbf{M}_A} c_{\nu q}
\end{aligned}$$

Note that for SCF methods (HF or DFT), $\boldsymbol{\sigma}_a^{A,\text{para/dia}} = 0$. To request the analysis, a valid GBW file must be given with the keyword `LocOrbGBW`, which can contain any orthonormal MOs in the same basis, e.g. canonical or localized, although the decomposition above assumes that the Brillouin condition is fulfilled and may be misleading if performed over NBOs for example. Orbital contributions over 0.01 ppm are printed – this is currently not user-configurable. The separate orbital pair contributions can also be printed using `Printlevel=3`. The following example input calculates HF and RI-MP2 shieldings for formaldehyde and decomposes them over Pipek–Mezey localized orbitals (note that the virtual orbitals are likely not well localized - `AHFB` would be better suited there). The second sub-calculation just prints the LMOs for convenient visualization with Avogadro.

```
! RI-MP2 NMR TightSCF RIJK pcSseg-1 cc-pwCVDZ/C def2/JK
%base "H2CO_NMR_PM"
```

(continues on next page)

(continued from previous page)

```
%loc
  LocMet PM    # Pipek-Mezey localization
  Occ    true  # localize both occupied
  Virt   true  # and virtual orbitals (better use AHFB for these!)
  T_core -1e5  # including the core
end
%eprnmr
  PrintLevel 3 # print orbital pair contributions (lots of output!)
  LocOrbGBW "H2CO_NMR_PM.loc" # use these orbitals
end
!xyzfile       # store the coordinates
*gzmt 0 1
  O
  C 1 1.2078
  H 2 1.1161 1 121.74
  H 2 1.1161 1 121.74 3 180
*
# read the localized orbitals and print them in the output
# for easy visualization with Avogadro
$new_job
! HF pcSseg-1 NoRI MORead NoIter PrintBasis PrintMOs
%moinp "H2CO_NMR_PM.loc"
*xyzfile 0 1 H2CO_NMR_PM.xyz
```

5.21.7 Treatment of Tau in Meta-GGA Functionals

For GIAO-based calculations with meta-GGAs, different options are available for the kinetic energy density τ. The current-independent τ_0 is not gauge-invariant. Ignoring the terms, which produce the gauge-dependence, leads to an ad-hoc gauge-invariant treatment (this was the default up to ORCA 5). A gauge-invariant definition τ_{MS}, containing an explicit dependence on the magnetic field, was proposed by Maximoff and Scuseria.[693] However, this ansatz produces unphysical paramagnetic contributions to the shielding tensor.[694] The last option, introduced by Dobson,[695] is gauge-invariant but requires the solution of the CP-SCF equations, even for pure density functionals. For a discussion and comparison of these alternatives see refs [696] (in the context of TDDFT) and [694, 697] (in the context of NMR shielding). Note that the calculated shieldings can differ substantially between the different approaches! Some other electronic structure programs use the MS ansatz by default, so be careful when comparing results between different codes. In ORCA the treatment of τ in GIAO-based calculations is chosed as follows

```
%eprnmr
  Tau 0      # gauge-variant
      GI     # ad-hoc gauge-invariant
      MS     # field-dependent, gauge-invariant version of Maximoff and Scuseria
      Dobson # (default) current density-dependent, gauge-invariant version of␣
↪Dobson
  end
```

5.21.8 Cartesian Index Conventions for EPR and NMR Tensors

The NMR shielding tensor $\boldsymbol{\sigma}$ and the EPR g and A tensors are in general nonsymmetric matrices. It is therefore important to know the conventions used with regard to their cartesian indices. These conventions stipulate the order of the vector–matrix–vector multiplications in the respective spin Hamiltonians. Unless stated otherwise, ORCA adopts the following conventions:

For the NMR shielding tensor the nuclear Zeeman Hamiltonian assumes the form:

$$H_I = -g_N \beta_N \mathbf{B}(\mathbf{1} - \boldsymbol{\sigma})\mathbf{I},$$

where $\mathbf{B}$ is the applied magnetic field vector.

For the EPR g and A tensors the EPR spin Hamiltonian assumes the form:

$$H_S = \beta_e \mathbf{BgS} + \mathbf{SAI}. \tag{5.96}$$

5.21.9 EPRNMR - keywords for magnetic properties

Calculation of EPR and NMR response properties can be requested in the `%eprnmr` input block. The individual flags are given below.

```
%eprnmr
  # Calculate the g-tensor using CP-KS theory
  gtensor true
  # Calculate and print one- and two-electron contributions to the g-tensor
  gtensor_1el2el true

  # Calculate the D-tensor
  DTensor so      # spin orbit part
          ss      # spin-spin part
          ssandso # both parts
  DSOC    qro     # quasi-restricted method; must be done with the keyword !uno
          pk      # Pederson-Khanna method.
                  # NOTE: both approximations are only valid for
                  # pure functionals (no HF exchange)
          cp      # coupled-perturbed method (default)
          cvw     # van W\"ullen method
  DSS     direct  # directly use the canonical orbitals for the spin density
          uno     # use spin density from UNOs

  PrintLevel n    # Amount of output (default 2)

  # whether to calculate and print the Euler angles via `orca_euler` if the
  # calculation of the g-tensor or the D-tensor is requested
  PrintEuler false

  # For the solution of the CP-SCF equations:
  Solver   Pople  # Pople solver (default)
           CG     # conjugate gradient
           DIIS   # DIIS type solver
  MaxIter  64     # maximum number of iterations
  MaxDIIS  10     # max. number of DIIS vectors (only DIIS)
  Tol      1e-3   # convergence tolerance
  LevelShift 0.05 # level shift for DIIS (ignored otherwise)

  Ori     CenterOfElCharge  # center of electronic charge
          CenterOfNucCharge # center of nuclear charge
          CenterOfSpinDens        # center of spin density
          CenterOfMass      # center of mass
          GIAO              # use the GIAO formalism (default)
          N                 # number of the atom to put the origin
          X,Y,Z             # explicit position of the origin
                            # in coordinate input units (Angstrom by default)

  # Calculate the NMR shielding tensor
  NMRShielding 1 # for chosen nuclei - specified with the Nuclei keyword
               2 # for all nuclei - equivalent to the 'NMR' simple input keyword

  # treatment of 1-electron integrals in the RHS of the CPSCF equations
  giao_1el = giao_1el_analytic   # analytical, default
             giao_1el_numeric    # numerical - for testing only

  # treatment of 2-electron integrals in the RHS of the CPSCF equations
```

(continues on next page)

(continued from previous page)

```
# various options for combination of approximations in Coulomb (J) and
# exact (HF) exchange (K) part. The default is the same as used for the SCF.
giao_2el = giao_2el_rijk        # RIJK approximation
           giao_2el_same_as_scf # use same scheme as in SCF
           giao_2el_analytic    # fully analytical
           giao_2el_rijonx      # RIJ approximation with analytical K
           giao_2el_cosjonx     # COSJ approximation with analytical K
           giao_2el_rijcosx     # RIJ approximation with COSX approximation
           giao_2el_cosjx       # COSJ approximation with COSX apprxomation
           giao_2el_exactjcosx  # exact J with COSX approximation
           giao_2el_exactjrik   # exact J with RIK approximation

# for g-tensor calculations using the SOMF-operator for the SOC
# treatment, the 2-electron contribution to the GIAO terms can be
# computed as well, but they take much more time and usually do not
# contribute significantly and therefore are disabled by default
do_giao_soc2el false

# treatment of tau in meta-GGA DFT - see above
Tau = Dobson # (default) Other options: 0, MS, GI

# use effective nuclear charges for the gauge correction to the A-tensor
# (this makes sense if an effective 1-electron SOC operator is used)
hfcgaugecorrection_zeff true

# calculate diamagnetic spin-orbit (DSO) integrals needed for the gauge
# correction to the A-tensor numerically (faster than the analytical way)
hfcgaugecorrection_numeric true

# Grid settings for the above: <0 means to use the DFT grid setting
hfcgaugecorrection_angulargrid -1
hfcgaugecorrection_intacc      -1
hfcgaugecorrection_prunegrid   -1
hfcgaugecorrection_bfcutoff    -1
hfcgaugecorrection_wcutoff     -1

Nuclei = all type { flags }
  # Calculate nuclear properties. Here the properties
  # for all nuclei of "type" are calculated ("type" is
  # an element name, for example Cu.)
  # Flags can be the following:
  #   aiso  - calculate the isotropic part of the HFC
  #   adip  - calculate the dipolar part of the HFC
  #   aorb  - 2nd order contribution to the HFC from SOC
  #   fgrad - calculate the electric field gradient
  #   rho   - calculate the electron density at the
  #           nucleus
  #   shift - NMR shielding tensor (orbital contribution)
  #   srot  - spin-rotation tensor
  #   ssdso - spin-spin coupling constants, diamagnetic spin orbit term
  #   sspso - spin-spin coupling constants, paramagnetic spin orbit term
  #   ssfc  - spin-spin coupling constants, Fermi contact term
  #   sssd  - spin-spin coupling constants, spin dipole term
  #   ssall - spin-spin coupling constants, calculate all above contributions

# In addition you can change several parameters
# e.g. for a different isotope.
Nuclei = all N { PPP=39.1, QQQ=0.5, III=1.0 };
    # PPP   - the HFC proportionality factor for this nucleus
    #         = ge*gN*betaE*betaN
    # QQQ   - the quadrupole moment for this nucleus
```

(continues on next page)

(continued from previous page)

```
    # III   - the spin for this nucleus
    # ist   - isotope
    # ssgn  - nuclear g-factor for spin-spin coupling
    #         and spin-rotation constants (overrides ist)

# For example:
# calculates the hyperfine coupling for all nitrogen atoms
Nuclei = all N { aiso, adip, fgrad, rho};

# calculates the spin-spin coupling constants for all carbon atoms
# assuming all carbon atoms are 13C
Nuclei = all C { ssall, ist = 13};

# You can also calculate properties for individual atoms
# as in the following example:
Nuclei = 1,5,8 { aiso, adip};

# NOTE: Counting starts with atom 1!
# WARNING: All the nuclei, mentioned in one line
# as above will be assigned the same parameters !

# For spin-spin coupling constants, a distance threshold is
# applied in the eprnmr block to restrict the number of couplings
# to be computed, given in Angstroms:
SpinSpinRThresh 5.0 # default

# Coupling can be restricted to certain element pairs
# (if they are added to the "Nuclei" list and are within RThresh).
# The syntax accepts multiple pairs of element symbols or
# atomic numbers or "*" as a wildcard
SpinSpinElemPairs {C C} {H *} {6 7} # default is {* *}, i.e. all

# Similarly, coupling can be restricted to certain atom pairs
# (if they are added to the "Nuclei" list and are within RThresh).
# The syntax accepts multiple pairs of indices (starting at 0!)
# or "*" as a wildcard
SpinSpinAtomPairs {1 0} {5 *} # default is {* *}, i.e. all

# whether to print reduced spin-spin coupling constants
PrintReducedCoupling false

end
```

5.22 Paramagnetic NMR Shielding Tensors

For systems with spin $S > 0$, the nuclear shielding contains a contribution which arises from the paramagnetism of the unpaired electrons.[3] This contribution is temperature-dependent and is called the "paramagnetic shielding" ($\boldsymbol{\sigma}^{\mathrm{p}}$). It adds to the temperature-indendent contribution to the shielding, also called the "orbital" contribution:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{orb}} + \boldsymbol{\sigma}^{\mathrm{p}}.$$

ORCA currently supports the calculation of $\boldsymbol{\sigma}^{\mathrm{p}}$ for systems whose paramagnetism can be described by the effective EPR spin Hamiltonian

$$H_S = \mathbf{SDS} + \beta_e \mathbf{BgS} + \mathbf{SAI}. \tag{5.97}$$

The theoretical background can be found in Refs. [698, 699]. We reproduce here the main equations.

[3] For a comprehensive review on paramagnetic NMR, see e.g. [701].

For a spin state described by Eq. (5.97), the paramagnetic shielding tensor is given by

$$\boldsymbol{\sigma}^{\mathrm{p}} = -\frac{\beta_e S(S+1)}{g_N \beta_N 3kT} \mathbf{gZA}, \tag{5.98}$$

where $\mathbf{Z}$ is a dimensionless 3×3 matrix which is determined by the ZFS and the temperature, as follows: Diagonalization of the ZFS Hamiltonian **SDS** yields energy levels E_λ and eigenstates $|S\lambda a\rangle$, where a labels degenerate states if E_λ is degenerate. Then $\mathbf{Z}$ is defined as $(i, j = x, y, z)$

$$Z_{ij} = \frac{3}{S(S+1)} \frac{1}{Q_0} \sum_\lambda e^{-E_\lambda/kT} \left[\sum_{a,a'} \langle S\lambda a|S_i|S\lambda a'\rangle \langle S\lambda a'|S_j|S\lambda a\rangle \right.$$
$$\left. +2kT \sum_{\lambda' \neq \lambda} \sum_{a,a'} \frac{\langle S\lambda a|S_i|S\lambda' a'\rangle \langle S\lambda' a'|S_j|S\lambda a\rangle}{E_{\lambda'} - E_\lambda} \right],$$

where $Q_0 = \sum_{\lambda',a} e^{-E_\lambda/kT}$ denotes the partition function. An important property of the $\mathbf{Z}$ matrix as defined above is that it goes to the identity matrix as $\mathbf{D}/kT$ goes to zero.

The orbital part of the shielding, $\boldsymbol{\sigma}^{\mathrm{orb}}$, is calculated in the same manner as for closed-shell molecules. It is available in ORCA for the unrestricted HF and DFT methods and for MP2 (see Section *MP2 Level Magnetic Properties* for more information on the latter).

The `orca_pnmr` tool uses Eq. *(5.101)* to calculate $\boldsymbol{\sigma}^{\mathrm{p}}$. Usage of `orca_pnmr` is described in Section *orca_pnmr*.

5.23 Spin-rotation Constants

Spin-rotation constant calculations are implemented using perturbation-dependent atomic orbitals following [700]. As given in eq. 34 of that reference, the spin-rotation tensor of nucleus K, $\mathbf{M}_K$, is related to the nuclear shielding tensor computed with GIAOs, $\boldsymbol{\sigma}_K^{\mathrm{GIAO}}$, and the diamagnetic part of the shielding tensor with the gauge origin set at that nucleus, $\boldsymbol{\sigma}_K^{\mathrm{dia}}(\mathbf{R}_K)$:

$$\mathbf{M}_K = 2\gamma_K \left(\boldsymbol{\sigma}_K^{\mathrm{GIAO}} - \boldsymbol{\sigma}_K^{\mathrm{dia}}(\mathbf{R}_K) \right) \mathbf{I}^{-1} - \mathbf{M}_K^{\mathrm{nuc}}$$

where $\mathbf{M}_K^{\mathrm{nuc}}$ is the nuclear component (given in eq. 14 of the reference), $\mathbf{I}$ is the inertia tensor, and γ_K is the nuclear magnetogyric ratio. Accordingly, upon requesting spin-rotation constants, ORCA automatically computes the NMR shieldings with GIAOs as well. Note that a complete list of possible keywords for the eprnmr module can be found in *EPRNMR - keywords for magnetic properties*.

The following input shows an example calculation of $\mathbf{M}(^{17}\mathrm{O})$ in $\mathrm{H_2{}^{12}C^{17}O}$:

```
! HF pcSseg-1 Mass2016 Bohrs
*xyz 0 1
     O        -0.00000000    -0.00000000     1.13863731 M=16.999131
     C        -0.00000000    -0.00000000    -1.14131773 M=12.0
     H        -0.00000000     1.76770755    -2.24076285
     H         0.00000000    -1.76770755    -2.24076285
*
%eprnmr
  Nuclei = all O {srot, ist=17}
end
```

Note

- The magnetogyric ratio used can be changed either by choosing the correct isotope via `ist`, or by providing the nuclear g-factor directly via `ssgn`.
- The masses used to compute the inertia tensor are independent of the chosen isotopes! The example above requests atomic masses of the most abundant isotope (via the `Mass2016` keyword) and explicitly specifies those of ^{12}C (which is the default) and ^{17}O.

5.24 Electron Paramagnetic Resonance (EPR) Parameters

Electron paramagnetic resonance probes the magnetic sublevels of the electronic ground state. In the `%eprnmr` input block, calculations of the parameters of a spin Hamiltonian of the form

$$H_S = \beta_e \mathbf{BgS} + \sum_N \mathbf{SA}_N \mathbf{I}_N + \mathbf{SDS} \tag{5.99}$$

can be requested. The properties are evaluated *via* analytical derivative/response theory methods. For the analytic response approach based on CC, please see *AUTOCI Response Properties via Analytic Derivatives*. For evaluation of spin Hamiltonian parameters using quasi-degenerate perturbation theory, you may be interested in *Magnetic Properties Through Quasi Degenerate Perturbation Theory* A complete list of possible keywords for the eprnmr module can be found in *EPRNMR - keywords for magnetic properties*. The cartesian index convention for EPR can be found in *Cartesian Index Conventions for EPR and NMR Tensors*.

5.24.1 The g-Tensor

The g-tensor determines the position of the signal in an EPR spectrum. It can be calculated as a second derivative of the energy and it is implemented as such in ORCA for the SCF methods (HF and DFT), CASSCF, as well as all-electron MP2 (or RI-MP2) and double-hybrid DFT, plus Coupled Cluster (CCSD and CCSD(T)).

Theory

The g-tensor has four contributions which may be evaluated in ORCA,

$$\mathbf{g} = g_e \mathbf{1} + \mathbf{g}^{\text{RMC}} + \mathbf{g}^{\text{DSO}} + \mathbf{g}^{\text{PSO}}. \tag{5.100}$$

These contributions are:

- The **spin Zeeman** contribution, which is isotropic and characterized by a simple constant, the free-electron g-value $g_e = 2.002319\ldots$

$$g^{\text{SZ}}_{\mu\nu} = \delta_{\mu\nu} g_e \tag{5.101}$$

  meaning that it doesn't require any computation. The remaining terms contribute to the deviation from the free-electron value, which is sometimes called the g-shift.

- The first-order **relativistic mass correction**, which is a usually rather small scalar-relativistic correction and can be evaluated as an expectation value over the spin density

$$\mathbf{g}^{\text{RMC}} = -\frac{\alpha^2 g_e}{2S} \sum_{k,l} P^{\alpha-\beta}_{kl} \left\langle \phi_k \left| \hat{T} \right| \phi_l \right\rangle. \tag{5.102}$$

- The **diamagnetic spin-orbit** (also called gauge correction) term, another first-order correction which is often small

$$g^{\text{DSO}}_{\mu\nu} = \frac{\alpha^2 g_e}{4S} \sum_{k,l} P^{\alpha-\beta}_{kl} \left\langle \phi_k \left| \sum_A \xi\left(r_A\right) \left[\mathbf{r}_A \mathbf{r}_O - \mathbf{r}_{A,\mu} \mathbf{r}_{O,\nu}\right] \right| \phi_l \right\rangle. \tag{5.103}$$

- The second-order **orbital Zeeman/SOC** contribution, usually the main source of deviation from the free-electron g-value in a molecule,

$$g^{\text{PSO}}_{\mu\nu} = -\frac{g_e}{2S} \sum_{k,l} \frac{\partial P^{\alpha-\beta}_{kl}}{\partial B_\mu} \left\langle \phi_k \left| h^{\text{SOC}}_\nu \right| \phi_l \right\rangle. \tag{5.104}$$

  Its calculation requires the solution of response equations to obtain the perturbed spin density, this is done for the magnetic field perturbation in ORCA. Precise details depend on the level of theory. At the SCF level, it is achieved by solving coupled-perturbed SCF equations (see *CP-SCF Options*). At the CASSCF level, the CP-CASSCF equations are solved (*CASSCF Linear Response*).

In these equations, S is the total spin, α the fine structure constant, $P^{\alpha-\beta}$ is the spin density matrix, ϕ is the basis set, $\hat{T}$ is the kinetic energy operator, $\xi(r_A)$ an approximate radial operator, h^{SOC} the spatial part of an effective one-electron spin-orbit operator and B_μ is a component of the magnetic field.

Basic usage

As an example, consider the following simple g-tensor job:

```
! BP86 Def2-SVP TightSCF g-tensor SOMF(1X)
* int 1 2
   O  0  0  0  0      0  0
   H  1  0  0  1.1056 0  0
   H  1  2  0  1.1056 109.62 0
*
```

The simplest way is to call the g-tensor property in the simple input line as shown above. It can also be specified in the `%eprnmr` block with `gtensor true`. `SOMF(1X)` defines the chosen spin-orbit coupling (SOC) operator.

The output looks like the following. It contains information on the contributions to the g-tensor (relativistic mass correction, diamagnetic spin-orbit term (= gauge-correction), paramagnetic spin-orbit term (= OZ/SOC)), the isotropic g-value and the orientation of the total tensor.

```
-------------------
ELECTRONIC G-MATRIX
-------------------

The g-matrix:
              2.0104321   -0.0031354   -0.0000000
             -0.0031354    2.0081968   -0.0000000
             -0.0000000   -0.0000000    2.0021275

Breakdown of the contributions
 gel          2.0023193    2.0023193    2.0023193
 gRMC        -0.0003174   -0.0003174   -0.0003174
 gDSO(tot)    0.0000808    0.0001539    0.0001515
 gPSO(tot)    0.0000449    0.0038301    0.0104898
             ----------   ----------   ----------
 g(tot)       2.0021275    2.0059858    2.0126431 iso=  2.0069188
 Delta-g     -0.0001917    0.0036665    0.0103238 iso=  0.0045995

 Orientation:
  X           0.0000000    0.5762906   -0.8172448
  Y           0.0000000    0.8172448    0.5762906
  Z           1.0000000   -0.0000000    0.0000000
```

g-tensor calculations at the SCF level are not highly demanding in terms of basis set size. Basis sets that give reliable SCF results (at least valence double-zeta plus polarization) usually also give reliable g-tensor results. For many molecules the Hartree-Fock approximation will give reasonable predictions. In a number of cases, however, it breaks down completely. DFT is more robust and the number of molecules where it fails is much smaller. Among the density functionals, the hybrid functionals seem to be the most accurate.

> **Note**
>
> There are different options for treatment of the spin-orbit coupling operator available in the program. The defaults should be quite reliable. Check out *The Spin-Orbit Coupling Operator* for more information.

Gauge origin treatment

The g-tensor is not gauge-independent. Without taking special precautions, the results will have an unphysical dependence on where the gauge origin is placed with respect to the molecule. Unless a fully invariant procedure (such as GIAOs) is used, this undesirable aspect is always present in the calculations.

Starting from ORCA 5.0, the default treatment for g-tensor calculations gauge is the **GIAO** approach (GIAO stands for "gauge including atomic orbitals"). GIAOs are available at the SCF level, RI-MP2 as well as Coupled Cluster up to CCSD(T) level. GIAOs are ***not*** currently available with CASSCF linear response and a common gauge origin must be provided in the `%eprnmr` block.

The GIAO one-electron integrals are done analytically by default whereas the treatment of the GIAO two-electron integrals is chosen to be same as for the SCF. The available options which can be set with `giao_1el / giao_2el` in the `%eprnmr` block can be found in section *EPRNMR - keywords for magnetic properties*. Concerning the computational time, for small systems, e.g. phenyl radical (41 electrons), the `rijk`-approximation is good to use for the SCF-procedures as well as the GIAO two-electron integrals. Going to larger systems, e.g. chlorophyll radical (473 electrons), the `rijcosx`-approximation reduces the computational time enormously. While the new default grid settings in ORCA 5.0 (`defgrid2`) should be sufficient in most cases, certain cases might need the use of `defgrid3`. Note that for the current implementation of CC-NMR/EPR, both `giao_1el` and `giao_2el` need to be evaluated fully analytically, which is also the default when CC-NMR/EPR is requested.

If the choice of the gauge origin is not outrageously poor, a **common gauge origin** often gives reasonable results for g-tensors (much more reasonable than in NMR, where it shouldn't be used). This is especially true with a large basis set. The origin may be modified with the keyword `ori` inside the `%eprnmr` input block. If you are using a common gauge origin, it is wise to check the sensitivity of the results with respect to origin location, especially when small g-shifts on the order of only a few hundred ppm are calculated.

Keywords

Table 5.13: `%eprnmr` block input keywords relevant for g-tensor calculations. For more GIAO-related options, see EPRNMR - keywords for magnetic properties

Keyword	Options	Description
`gtensor`	`true/false`	Calculate the g-tensor
`gtensor_1e`	`true/false`	Print the 1- and 2-electron contributions to the g-tensor
`ori`	`GIAO`	use the GIAO formalism (default)
	`CenterOfElC`	Common gauge origin at center of electronic charge
	`CenterOfNuc`	Common gauge origin at center of nuclear charge
	`CenterOfSpi`	Common gauge origin at center of spin density
	`CenterOfMas`	Common gauge origin at center of mass
	`N`	Common gauge origin at atom `N`
	`x, y, z`	Common gauge origin at position `x, y, z` (in coordinate input units, default Angstrom)
`do_giao_s`	`true/false`	whether to include the 2-el contribution to GIAO term from the SOMF operator (usually small but expensive, disabled by default)

5.24.2 Hyperfine Coupling

Hyperfine couplings characterize the interaction between the electronic spin and the spin of a nucleus in the molecule. In typical EPR spectra, they lead to a splitting of the signal.

Theory

The hyperfine coupling has four contributions which may be evaluated in ORCA,

$$\mathbf{A}_N = a_{iso}\mathbf{1} + \mathbf{A}^{\text{dip}} + \mathbf{A}^{\text{orb}} + \mathbf{A}^{\text{GC}}. \tag{5.105}$$

These contributions are:

- The isotropic **Fermi contact** term that arises from finite spin density on the nucleus under investigation. It is calculated for nucleus N from:

$$a_{\text{iso}}(N) = \left(\frac{4}{3}\pi \langle S_z \rangle^{-1}\right) g_e g_N \beta_e \beta_N \rho\left(\vec{R}_N\right) \tag{5.106}$$

  Here, $\langle S_z \rangle$ is the expectation value of the z-component of the total spin, g_e and g_N are the electron and nuclear g-factors and β_e and β_N are the electron and nuclear magnetons respectively. $\rho\left(\vec{R}_N\right)$ is the spin density at the nucleus. The proportionality factor $P_N = g_e g_N \beta_e \beta_N$ is commonly used and has the dimensions MHz bohr^3 in ORCA.

- The **spin dipole** term that arises from the through-space dipole-dipole interaction of the magnetic nucleus with the magnetic moment of the electron. It is also calculated as an expectation value over the spin density as:

$$A^{\text{dip}}_{\mu\nu}(N) = P_N \sum_{kl} \rho_{kl} \left\langle \phi_k \left| r_N^{-5} \left(3\vec{r}_{N\mu}\vec{r}_{N\nu} - \delta_{\mu\nu} r_N^2\right) \right| \phi_l \right\rangle \tag{5.107}$$

  where ρ is the spin-density matrix and $\vec{r}_N$ is a vector of magnitude r_N that points from the nucleus in question to the electron ($\{\phi\}$ is the set of basis functions).

- The second-order **spin-orbit** contribution, which arises as a cross-term between the spin-orbit and nucleus-orbit coupling operators. It requires the solution of coupled-perturbed SCF equations and is consequently more computationally demanding. The contribution can be written as:

$$A^{\text{orb}}_{\mu\nu}(N) = -\frac{1}{2S} P_N \sum_{kl} \frac{\partial \rho_{kl}}{\partial I_\nu} \left\langle \phi_k \left| h^{\text{SOC}}_\mu \right| \phi_l \right\rangle \tag{5.108}$$

  The derivative of the spin density is computed by solving the coupled-perturbed SCF equations with respect to the nucleus-orbit coupling perturbation, which is represented by the operator

$$h^{\text{NOC}}_\nu(N) = \sum_i r_{iA}^{-3} l^{(N)}_{i,\nu} \tag{5.109}$$

  where the sum is over electrons and N is the nucleus in question.

- The **gauge-correction** contribution (sometimes also called diamagnetic). This term is often small. However, it is needed in order to get exactly gauge-invariant results. We recently showed that the gauge correction can become crucial in the long-distance limit between the nuclear spin and the electron spin. This is relevant for pseudocontact NMR chemical shifts (PCS).[702]

> **Note**
>
> Second-order HFCs can be quite significant for heavier nuclei and are certainly good to include for transition metal complexes. Available treatments of the spin-orbit coupling operator are described under *The Spin-Orbit Coupling Operator*.

Basic usage

Hyperfine and quadrupole couplings can be requested in the `%eprnmr` input block. Since there may be several nuclei that you are interested in, the input is relatively sophisticated.

An example how to calculate the hyperfine and field gradient tensors for the CN radical is given below:

```
! PBE0 def2-MSVP TightSCF
* int 0 2
   C  0  0  0  0     0  0
   N  1  0  0  1.170 0  0
*
%eprnmr  Nuclei = all C { aiso, adip }
         Nuclei = 2 { aiso, adip, fgrad }
         end
```

In this example, the hyperfine tensor is calculated for all carbon atoms and atom 2, which is nitrogen in this case. (Additionally, the electric field gradient is calculated for nitrogen.)

Note

- Counting of atom numbers starts from 1.
- Beware - all nuclei mentioned in one line will be assigned the same isotopic mass! If several different nuclei are desired (such as C and H), there has to be a new line for each of them.
- You have to specify the `Nuclei` statement *after* the definition of the atomic coordinates or the program will not figure out what is meant by "`all`".

The output looks like the following. It contains detailed information about the individual contributions to the hyperfine couplings, its orientation, its eigenvalues, the isotropic part and (if requested) also the quadrupole coupling tensor.

```
----------------------------------------
ELECTRIC AND MAGNETIC HYPERFINE STRUCTURE
----------------------------------------

 ------------------------------------------------------------
 Nucleus   0C : A:ISTP=   13 I=  0.5 P=134.1903 MHz/au**3
                Q:ISTP=   13 I=  0.5 Q=  0.0000 barn
 ------------------------------------------------------------
 Raw HFC matrix (all values in MHz):
 -----------------------------------
               695.8952              0.0000              -0.0000
                 0.0000            543.0617              -0.0000
                -0.0000             -0.0000             543.0617

 A(FC)         594.0062            594.0062             594.0062
 A(SD)         -50.9445            -50.9445             101.8890
             ----------          ----------           ----------
 A(Tot)        543.0617            543.0617             695.8952    A(iso)=  594.
↪0062
 Orientation:
  X           0.0000000           0.0000000           -1.0000000
  Y          -0.8111216          -0.5848776           -0.0000000
  Z          -0.5848776           0.8111216            0.0000000

Notes:  (1) The A matrix conforms to the "SAI" spin Hamiltonian convention.
        (2) Tensor is right-handed.

 ------------------------------------------------------------
 Nucleus   1N : A:ISTP=   14 I=  1.0 P= 38.5677 MHz/au**3
```

(continues on next page)

(continued from previous page)

```
                Q:ISTP=   14 I=  1.0 Q=  0.0204 barn
 ------------------------------------------------------------
 Raw HFC matrix (all values in MHz):
 -----------------------------------
                13.2095              -0.0000               0.0000
                -0.0000             -45.6036              -0.0000
                 0.0000              -0.0000             -45.6036

 A(FC)          -25.9993             -25.9993             -25.9993
 A(SD)           39.2088             -19.6044             -19.6044
              ----------           ----------           ----------
 A(Tot)          13.2095             -45.6036             -45.6036    A(iso)=  -25.
↪9993
 Orientation:
  X           1.0000000            0.0000000           -0.0000000
  Y          -0.0000000            0.9996462            0.0265986
  Z           0.0000000           -0.0265986            0.9996462

Notes:  (1) The A matrix conforms to the "SAI" spin Hamiltonian convention.
        (2) Tensor is right-handed.

 Raw EFG matrix (all values in a.u.**-3):
 ----------------------------------
            -0.1832      -0.0000       0.0000
            -0.0000       0.0916       0.0000
             0.0000       0.0000       0.0916

 V(El)       0.6468       0.6468      -1.2935
 V(Nuc)     -0.5551      -0.5551       1.1103
         ----------   ----------   ----------
 V(Tot)      0.0916       0.0916      -0.1832
 Orientation:
  X      -0.0000003    0.0000002    1.0000000
  Y       0.9878165    0.1556229    0.0000003
  Z      -0.1556229    0.9878165   -0.0000002

Note: Tensor is right-handed


 Quadrupole tensor eigenvalues (in MHz;Q= 0.0204 I=  1.0)
  e**2qQ            =    -0.880 MHz
  e**2qQ/(4I*(2I-1))=    -0.220 MHz
  eta               =     0.000
  NOTE: the diagonal representation of the SH term I*Q*I = e**2qQ/(4I(2I-1))*[-(1-
↪eta),-(1+eta),2]
```

If also EPR g-tensor or D-tensor calculations (see next section) are carried out in the same job, ORCA automatically prints the orientation between the hyperfine/quadrupole couplings and the molecular g- or D-tensor. For more information on this see section *orca_euler*.

Note

For heavy nuclei you may want to consider the possibility of relativistic effects. Scalar relativistic effects can be handled with several quasi-relativistic Hamiltonians in ORCA. An overview of the possibilities and some recommendations can be found in *Relativistic Calculations*. Note that relativistic calculations may have special requirements on basis sets. In relativistic property calculations, you should be aware of the importance of picture change corrections (see *Picture-Change Effects*). In quasi-relativistic calculations with DFT, one should also be very cautious about accuracy of the numerical integration, especially for heavier (transition metal) nuclei.

Note

For the calculation of HFCCs using DLPNO-CCSD, it is recommended to use the tailored truncation settings `!DLPNO-HFC1` or `!DLPNO-HFC2` in the simple keyword line which correspond to the "Default1" and "Default2" setting in Ref. [703].

If you wish to extract the *A* tensor for an oligonuclear transition metal complex, the `A(iso)` value in the output can be processed according to the method described in ref. [704].

A note on basis sets

For hyperfine (and quadrupole) couplings, standard basis sets designed for energies and geometry optimizations and are often not satisfactory (especially for atoms heavier than Ne). You should look into a basis set capable of describing the electronic structure near the nucleus. One option is to use a basis set tailored towards "core-property" calculations. The following are good options:

- `EPR-II` basis of Barone and co-workers: It is only available for a few light atoms (H, B, C, N, O, F) and is essentially of double-zeta plus polarization quality with added flexibility in the core region, which should give reasonable results.
- `IGLO-II` and `IGLO-III` bases of Kutzelnigg and co-workers: They are fairly accurate but also only available for some first and second row elements.
- CP basis: They are accurate for first row transition metals as well.
- uncontracted Partridge basis: They are general purpose HF-limit basis sets and will probably be too expensive for routine use, but are very useful for calibration purposes.
- the `pcH` basis sets by Jensen tailored towards hyperfine coupling calculations, also with limited availability (see *Jensen Basis Sets*).

If ORCA does not yet have a dedicated basis set for your element, you will likely have to tailor the basis set to your needs. You can do this by manually adding s-type primitives with large exponents, or by decontracting core parts of the basis set (see *Basis Sets* for description of the decontraction feature). You can start by examining the basis set you already have - if you add the statement `Print[p_basis] 2` in the `%output` block (or `PrintBasis` in the simple input line) the program will print the basis set in input format (for the basis block). You can then add or remove primitives, uncontract core functions, etc. For example, here is a printout of the carbon basis DZP in input format:

```
# Basis set for element : C
  NewGTO 6
  s 5
   1  3623.8613000000      0.0022633312
   2   544.0462100000      0.0173452633
   3   123.7433800000      0.0860412011
   4    34.7632090000      0.3022227208
   5    10.9333330000      0.6898436475
  s 1
   1     3.5744765000      1.0000000000
  s 1
    1    0.5748324500      1.0000000000
  s 1
    1    0.1730364000      1.0000000000
  p 3
    1    9.4432819000      0.0570590790
    2    2.0017986000      0.3134587330
    3    0.5462971800      0.7599881644
  p 1
    1    0.1520268400      1.0000000000
  d 1
```

(continues on next page)

(continued from previous page)

```
  1    0.8000000000      1.0000000000
end;
```

Reading this information is relatively straightforward. For example: the "`s 5`" stands for the angular momentum and the number of primitives in the first basis function. The following five lines then contain the number of the primitive, the exponent and the contraction coefficient (unnormalized) for each primitive in the function.

Remember that if you add very steep functions, you *must* increase the size of the integration grid if your calculation uses DFT! Otherwise, your results will be inaccurate. To some extent, the program now automatically adapts the grids when it detects steep functions. If you further need to increase integration accuracy, you could globally increase the radial grid size by increasing `IntAcc` in the `Method` block, or increase integration accuracy for individual atoms only, with the latter option being less expensive. More detail can be found in section *Other Details and Options*. In the present example the changes caused by larger basis sets in the core region and more accurate integration are relatively modest – on the order of 3%. This is, however, still significant if you are a little puristic.

Hyperfine Coupling Keywords

Warning

All nuclei specified on one line will be assigned with the same isotopic mass (and other parameters)!

Table 5.14: `%eprnmr` block input keywords relevant for HFC calculations

Key-word	Selection	What to do	Description
`nuclei`	`all <type>`		Selects all nuclei of element `<type>`
	`i1,i2,i3...`		Selects atoms `i1`, `i2`, `i3`... (starts at 1!)
		`{ aiso }`	Calculates the isotropic Fermi-contact contribution
		`{ adip }`	Calculates the spin-dipole contribution
		`{ aorb }`	Calculates the spin-orbit contribution
		`{ ist=<N> }`	Selects isotope `<N>` for selection
		`{ PPP=<N> }`	Sets HFC proportionality factor $g_e g_N \beta_E \beta_N$ to `<N>` for selection
		`{ III=<N> }`	Sets nuclear spin to `<N>` for selection
		`{ aiso, adip, aorb }`	example requesting multiple actions for selection

Table 5.15: `%eprnmr` block: possible tweaks for the diamagnetic gauge correction. Unless you have a special reason, it is unlikely that you need to use these settings.

Keyword	Options	Description
`hfcgaugecorrection_zeff`	`true/false`	use effective nuclear charges for the gauge correction
`hfcgaugecorrection_numeric`	`true/false`	calculate DSO integrals numerically (faster)
`hfcgaugecorrection_angulargri`	`-1`	< 0 means to use the DFT grid setting
`hfcgaugecorrection_intacc`	`-1`	< 0 means to use the DFT grid setting
`hfcgaugecorrection_prunegrid`	`-1`	< 0 means to use the DFT grid setting
`hfcgaugecorrection_bfcutoff`	`-1`	< 0 means to use the DFT grid setting
`hfcgaugecorrection_wcutoff`	`-1`	< 0 means to use the DFT grid setting

5.24.3 Zero-Field Splitting

The zero-field splitting (ZFS) is typically the leading term in the Spin-Hamiltonian (SH) for transition metal complexes with a total ground state spin S>1/2 (for reviews and references see chapter *Publications Related to ORCA*). Its effect is to introduce a splitting of the $2S+1$ M_S sublevels, which are exactly degenerate at the level of the Born-Oppenheimer Hamiltonian, even in the absence of an external magnetic field.

Theory

There are two important contributions to zero-field splitting [705]:

- A first order term arising from the direct **spin-spin** interaction

$$D_{KL}^{\mathrm{SS}} = \frac{1}{2}\frac{\alpha^2}{S\left(2S-1\right)}\left\langle 0SS\left|\sum_i\sum_{j\neq i}\frac{r_{ij}^2\delta_{KL}-3\left(\mathbf{r}_{ij}\right)_K\left(\mathbf{r}_{ij}\right)_L}{r_{ij}^5}\left\{2\hat{s}_{zi}\hat{s}_{zj}-\hat{s}_{xi}\hat{s}_{xj}-\hat{s}_{yi}\hat{s}_{yj}\right\}\right|0SS\right\rangle \tag{5.110}$$

  where K,L =x,y,z, α is the fine structure constant ($\approx$ 1/137 in atomic units), $\mathbf{r}_{ij}$ is the electronic distance vector with magnitude r_{ij} and $\hat{s}_i$ is the spin-vector operator for the i'th electron. $|0SS\rangle$ is the exact ground state eigenfunction of the Born-Oppenheimer Hamiltonian with total spin S and projection quantum number $M_S = S$. Since the spin-spin interaction is of first order, its evaluation presents no particular difficulties.

- A second-order **spin-orbit** contribution, which is substantially more complicated. Under the assumption that the spin-orbit coupling (SOC) operator can to a good approximation be represented by an effective one-electron operator ($\hat{H}_{\mathrm{SOC}} = \sum_i \hat{\mathbf{h}}_i^{\mathrm{SOC}}\hat{\mathbf{s}}_i$), ref [656] has derived the following sum-over-states (SOS) equations for the SOC contribution to the ZFS tensor:

$$D_{KL}^{\mathrm{SOC}-(0)} = -\frac{1}{S^2}\sum_{b(S_b=S)}\Delta_b^{-1}\left\langle 0^{SS}\left|\sum_i\hat{h}_i^{K;\mathrm{SOC}}\hat{s}_{i,0}\right|b^{SS}\right\rangle\left\langle b^{SS}\left|\sum_i\hat{h}_i^{L;\mathrm{SOC}}\hat{s}_{i,0}\right|0^{SS}\right\rangle \tag{5.111}$$

$$D_{KL}^{\mathrm{SOC}-(-1)} = -\frac{1}{S\left(2S-1\right)}\sum_{b(S_b=S-1)}\Delta_b^{-1}\left\langle 0^{SS}\left|\sum_i\hat{h}_i^{K;\mathrm{SOC}}\hat{s}_{i,+1}\right|b^{S-1S-1}\right\rangle\left\langle b^{S-1S-1}\left|\sum_i\hat{h}_i^{L;\mathrm{SOC}}\hat{s}_{i,-1}\right|0^{SS}\right\rangle \tag{5.112}$$

$$D_{KL}^{\mathrm{SOC}-(+1)} = -\tfrac{1}{(S+1)(2S+1)}\cdot \sum_{b(S_b=S+1)}\Delta_b^{-1}\left\langle 0^{SS}\left|\sum_i\hat{h}_i^{K;\mathrm{SOC}}\hat{s}_{i,-1}\right|b^{S+1S+1}\right\rangle\left\langle b^{S+1S+1}\left|\sum_i\hat{h}_i^{L;\mathrm{SOC}}\hat{s}_{i,+1}\right|0^{SS}\right\rangle \tag{5.113}$$

  Here the one-electron spin-operator for electron i has been written in terms of spherical vector operator components $s_{i,m}$ with $m = 0, \pm 1$ and $\Delta_b = E_b - E_0$ is the excitation energy to the excited state multiplet $|b^{SS}\rangle$ (all M_S components are degenerate at the level of the BO Hamiltonian).

Note

The second-order spin-orbit term may be evaluated using the above sum-over-states formulation in the context of wavefunction theory methods. The following section focuses on a formulation based on SCF analytical derivative/response theory. You may also be interested in QDPT, which is more reliable particularly in systems where zero-field splitting is large.

Note

Available treatments of the spin-orbit coupling operator are described under *The Spin-Orbit Coupling Operator*.

Basic usage

For example, consider the following job on a hypothetical Mn(III)-complex.

```
! BP86 def2-SVP SOMF(1X)

%eprnmr DTensor  ssandso
        DSOC     cp  # qro, pk, cvw
        DSS      uno # direct
        end

* int  1 5
Mn 0 0 0  0 0 0
 O 1 0 0  2.05   0   0
 O 1 2 0  2.05  90   0
 O 1 2 3  2.05  90 180
 O 1 2 3  2.05 180   0
 F 1 2 3  1.90  90  90
 F 1 2 3  1.90  90 270
 H 2 1 6  1.00 127   0
 H 2 1 6  1.00 127 180
 H 3 1 6  1.00 127   0
 H 3 1 6  1.00 127 180
 H 4 1 6  1.00 127   0
 H 4 1 6  1.00 127 180
 H 5 1 6  1.00 127   0
 H 5 1 6  1.00 127 180
*
```

The output documents the individual contributions to the D-tensor, which also contains (unlike the g-tensor) contributions from spin-flip terms.

Note

There are four different variants of the SOC-contribution, which alone should demonstrate that this is a difficult property. The options are:

- The QRO method is fully documented[360] and is based on a theory developed earlier.[656] The QRO method is reasonable but somewhat simplistic and is superseded by the CP method described below.
- The Pederson-Khanna model was brought forward in 1999 from qualitative reasoning.[706] It also contains incorrect prefactors for the spin-flip terms. We have nevertheless implemented the method for comparison. In the original form it is only valid for local functionals. In ORCA it is extended to hybrid functionals and HF.
- The default coupled-perturbed method is a generalization of the DFT method for ZFSs; it uses revised prefactors for the spin-flip terms and solves a set of coupled-perturbed equations for the SOC perturbation.

Therefore it is valid for hybrid functionals. It has been described in detail.[707]

The present implementation in ORCA is valid for HF, DFT and hybrid DFT.

The DSS part is an expectation value that involves the spin density of the system. In detailed calibration work[708] it was found that the spin-unrestricted DFT methods behave somewhat erratically and that much more accurate values were obtained from open-shell spin-restricted DFT. Therefore the "UNO" option allows the calculation of the SS term with a restricted spin density obtained from the singly occupied unrestricted natural orbitals.

The DSS part contains an erratic self-interaction term for UKS/UHF wavefunction and canonical orbitals. Thus, `UNO` is recommended for these types of calculations.[709] If the option `DIRECT` is used nevertheless, ORCA will print a warning in the respective part of the output.

Note

In case that D-tensor is calculated using the correlated wave function methods such as (DLPNO-/LPNO-)CCSD, one should not use `DSS=UNO` option.

Detailed theory of the coupled-perturbed formulation

In 2007, we have developed a procedure that makes the ZFS calculation compatible with the language of analytic derivatives.[707] Perhaps the most transparent route is to start from the exact solutions of the Born-Oppenheimer Hamiltonian. To this end, we look at the second derivative of the ground state energy ($E = \left\langle 0^{SS} \middle| \hat{H} \middle| 0^{SS} \right\rangle$) with respect to a spin-dependent one-electron operator of the general form:

$$\hat{h}^{K;(m)} = x_K^{(m)} \sum_{pq} h_{pq}^K \hat{S}_{pq}^{(m)} \tag{5.114}$$

Where h_{pq}^K is the matrix of the K'th component of the spatial part of the operator (assumed to be imaginary Hermitian as is the case for the spatial components of the SOC operator) and $\hat{S}_{pq}^{(m)}$ is the second quantized form of the spin vector operator ($m = 0, \pm 1$). The quantity $x_K^{(m)}$ is a formal perturbation parameter. Using the exact eigenfunctions of the BO operator, the first derivative is:

$$\left. \frac{\partial E}{\partial x_K^{(m)}} \right|_{x_K^{(m)}=0} = \sum_{pq} h_{pq}^K P_{pq}^{(m)} \tag{5.115}$$

With the components of the spin density:

$$P_{pq}^{(m)} = \left\langle 0^{SS} | \hat{S}_{pq}^{(m)} | 0^{SS} \right\rangle \tag{5.116}$$

The second derivative with respect to a second component for $m' = -m$ is:

$$\left. \frac{\partial^2 E}{\partial x_K^{(m)} \partial x_L^{(-m)}} \right|_{x_K^{(m)}=x_L^{(-m)}=0} = \sum_{pq} h_{pq}^K \frac{\partial P_{pq}^{(m)}}{x_L^{(-m)}} \tag{5.117}$$

The derivative of the spin density may be written as:

$$\frac{\partial P_{pq}^{(m)}}{x_L^{(-m)}} = \left\langle 0_L^{SS(-m)} | \hat{S}_{pq}^{(m)} | 0^{SS} \right\rangle + \left\langle 0^{SS} | \hat{S}_{pq}^{(m)} | 0_L^{SS(-m)} \right\rangle \tag{5.118}$$

Expanding the perturbed wavefunction in terms of the unperturbed states gives to first order:

$$\left| 0_L^{SS(-m)} \right\rangle = -\sum n \neq 0 \Delta_n^{-1} |n\rangle \left\langle n \middle| \hat{h}^{L;(-m)} \middle| 0^{SS} \right\rangle \tag{5.119}$$

Where $|n\rangle$ is any of the $\left|b^{S'M'}\right\rangle$. Thus, one gets:

$$\frac{\partial^2 E}{\partial x_K^{(m)} \partial x_L^{(-m)}} = \sum_{pq} h_{pq}^K \frac{\partial P_{pq}^{(m)}}{x_L^{(-m)}} \tag{5.120}$$

$$= -\sum_{n\neq 0} \Delta_n^{-1} \left[\left\langle 0^{SS} | \hat{h}^{L;(-m)} | n \right\rangle \left\langle n | \hat{h}^{K;(m)} | 0^{SS} \right\rangle + \left\langle 0^{SS} | \hat{h}^{K;(m)} | n \right\rangle \left\langle n | \hat{h}^{L;(-m)} | 0^{SS} \right\rangle \right] \tag{5.121}$$

The equality holds for exact states. For approximate electronic structure treatments, the analytic derivative approach is more attractive since an infinite sum over states can never be performed in practice and the calculation of analytic derivative is computationally less demanding than the calculation of excited many electron states.

Using eq. (5.120), the components of the SOC-contribution to the **D**-tensor are reformulated as

$$D_{KL}^{\text{SOC}-(0)} = \frac{1}{2S^2} \sum_{pq} h_{pq}^{K;\text{SOC}} \frac{\partial P_{pq}^{(0)}}{\partial x_L^{(0)}} \tag{5.122}$$

$$D_{KL}^{\text{SOC}-(-1)} = \frac{1}{S\left(2S-1\right)} \sum_{pq} h_{pq}^{K;\text{SOC}} \frac{\partial P_{pq}^{(+1)}}{\partial x_L^{(-1)}} \tag{5.123}$$

$$D_{KL}^{\text{SOC}-(+1)} = \frac{1}{\left(S+1\right)\left(2S+1\right)} \sum_{pq} h_{pq}^{K;\text{SOC}} \frac{\partial P_{pq}^{(-1)}}{\partial x_L^{(+1)}} \tag{5.124}$$

These are general equations that can be applied together with any non-relativistic or scalar relativistic electronic structure method that can be cast in second quantized form. Below, the formalism is applied to the case of a self-consistent field (HF, DFT) reference state.

For DFT or HF ground states, the equations are further developed as follows:

The SCF energy is:

$$E_{\text{SCF}} = V_{\text{NN}} + \left\langle \mathbf{Ph}^+ \right\rangle + \frac{1}{2} \int \int \frac{\rho\left(\mathbf{r}_1\right) \rho\left(\mathbf{r}_2\right)}{|\mathbf{r}_1 - \mathbf{r}_2|} d\mathbf{r}_1 d\mathbf{r}_2 - \frac{1}{2} a_{\text{X}} \sum_{\mu\nu\kappa\tau\sigma} P_{\mu\kappa}^{\sigma} P_{\nu\tau}^{\sigma} \left(\mu\nu|\kappa\tau\right) + c_{\text{DF}} E_{\text{XC}} \left[\rho_\alpha, \rho_\beta\right] \tag{5.125}$$

Here V_{NN} is the nuclear repulsion energy and $h_{\mu\nu}$ is a matrix element of the one-electron operator which contains the kinetic energy and electron-nuclear attraction terms ($\langle \mathbf{ab} \rangle$ denotes the trace of the matrix product **ab**). As usual, the molecular spin-orbitals ψ_p^σ are expanded in atom centered basis functions ($\sigma = \alpha, \beta$):

$$\psi_p^\sigma\left(\mathbf{r}\right) = \sum_\mu c_{\mu p}^\sigma \phi_\mu\left(\mathbf{r}\right) \tag{5.126}$$

with MO coefficients $c_{\mu p}^\sigma$. The two-electron integrals are defined as:

$$\left(\mu\nu|\kappa\tau\right) = \int \int \phi_\mu\left(\mathbf{r}_1\right) \phi_\nu\left(\mathbf{r}_1\right) r_{12}^{-1} \phi_\kappa\left(\mathbf{r}_2\right) \phi_\tau\left(\mathbf{r}_2\right) d\mathbf{r}_1 d\mathbf{r}_2 \tag{5.127}$$

The mixing parameter a_{X} controls the fraction of Hartree-Fock exchange and is of a semi-empirical nature. $E_{\text{XC}}\left[\rho_\alpha, \rho_\beta\right]$ represent the exchange-correlation energy. The parameter c_{DF} is an overall scaling factor that allows one to proceed from Hartree-Fock theory ($a_{\text{X}} = 1, c_{\text{DF}} = 0$) to pure DFT ($a_{\text{X}} = 0, c_{\text{DF}} = 1$) to hybrid DFT ($0 < a_{\text{X}} < 1, c_{\text{DF}} = 1$). The orbitals satisfy the spin-unrestricted SCF equations:

$$F_{\mu\nu}^\sigma = h_{\mu\nu} + \sum_{\kappa\tau} P_{\kappa\tau}\left(\mu\nu|\kappa\tau\right) - a_{\text{X}} P_{\kappa\tau}^\sigma \left(\mu\kappa|\nu\tau\right) + c_{\text{DF}} \left(\mu|V_{XC}^\alpha|\nu\right) \tag{5.128}$$

With $V_{XC}^\sigma = \frac{\delta E_{XC}}{\delta \rho_\sigma(\mathbf{r})}$ and $P_{\mu\nu} = P_{\mu\nu}^\alpha + P_{\mu\nu}^\beta$ being the total electron density. For the SOC perturbation it is customary to regard the basis set as perturbation independent. In a spin-unrestricted treatment, the first derivative is:

$$\frac{\partial E_{\text{SCF}}}{\partial x_K^{(m)}} = \sum_{i_\alpha} \left(i_\alpha | h^K s_m | i_\alpha\right) + \sum_{i_\beta} \left(i_\beta | h^K s_m | i_\beta\right) = 0 \tag{5.129}$$

For the second derivative, the perturbed orbitals are required. However, in the presence of a spin-dependent perturbation they can no longer be taken as pure spin-up or spin-down orbitals. With respect to the L'th component of the perturbation for spin-component m, the orbitals are expanded as:

$$\psi_i^{\alpha;(m)L}(\mathbf{r}) = \sum_{a_\alpha} U_{a_\alpha i_\alpha}^{(m);L} \psi_a^\alpha(\mathbf{r}) + \sum_{a_\beta} U_{a_\beta i_\alpha}^{(m);L} \psi_a^\beta(\mathbf{r}) \tag{5.130}$$

$$\psi_i^{\beta;(m)L}(\mathbf{r}) = \sum_{a_\alpha} U_{a_\alpha i_\beta}^{(m);L} \psi_a^\alpha(\mathbf{r}) + \sum_{a_\beta} U_{a_\beta i_\beta}^{(m);L} \psi_a^\beta(\mathbf{r}) \tag{5.131}$$

Since the matrix elements of the spin-vector operator components are purely real and the spatial part of the SOC operator has purely imaginary matrix elements, it follows that the first order coefficients are purely imaginary. The second derivative of the total SCF energy becomes:

$$\begin{aligned}\frac{\partial^2 E_{\text{SCF}}}{\partial x_K^{(m)} \partial x_L^{(-m)}} = &\sum_{i_\alpha a_\alpha} \left\{ U_{a_\alpha i_\alpha}^{(-m);L*} \left(a_\alpha | h^K s_m | i_\alpha\right) + U_{a_\alpha i_\alpha}^{(-m);L} \left(i_\alpha | h^K s_m | a_\alpha\right) \right\} \\ &+ \sum_{i_\alpha a_\beta} \left\{ U_{a_\beta i_\alpha}^{(-m);L*} \left(a_\beta | h^K s_m | i_\alpha\right) + U_{a_\beta i_\alpha}^{(-m);L} \left(i_\alpha | h^K s_m | a_\beta\right) \right\} \\ &+ \sum_{i_\beta a_\alpha} \left\{ U_{a_\alpha i_\beta}^{(-m);L*} \left(a_\alpha | h^K s_m | i_\beta\right) + U_{a_\alpha i_\beta}^{(-m);L} \left(i_\beta | h^K s_m | a_\alpha\right) \right\} \\ &+ \sum_{i_\beta a_\beta} \left\{ U_{a_\beta i_\beta}^{(-m);L*} \left(a_\beta | h^K s_m | i_\beta\right) + U_{a_\beta i_\beta}^{(-m);L} \left(i_\beta | h^K s_m | a_\beta\right) \right\}\end{aligned} \tag{5.132}$$

Examination of the three cases $m = 0, \pm 1$ leads to the following equations for the **D**-tensor components:

$$D_{KL}^{(0)} = -\frac{1}{4S^2} \sum_{\mu\nu} \frac{\partial P_{\mu\nu}^{(0)}}{\partial x_L^{(0)}} \left(\mu | h^{K;\text{SOC}} | \nu\right) \tag{5.133}$$

$$D_{KL}^{(+1)} = \frac{1}{2(S+1)(2S+1)} \sum_{\mu\nu} \left(\mu | h^{K;\text{SOC}} | \nu\right) \frac{\partial P_{\mu\nu}^{(-1)}}{\partial x_L^{(+1)}} \tag{5.134}$$

$$D_{KL}^{(-1)} = \frac{1}{2S(2S-1)} \sum_{\mu\nu} \left(\mu | h^{K;\text{SOC}} | \nu\right) \frac{\partial P_{\mu\nu}^{(+1)}}{\partial x_L^{(-1)}} \tag{5.135}$$

Where a special form of the perturbed densities has been chosen. They are given in the atomic orbital basis as:

$$\frac{\partial P_{\mu\nu}^{(0)}}{\partial x_L^{(0)}} = \sum_{i_\alpha a_\alpha} U_{a_\alpha i_\alpha}^{(0);L} c_{\mu i}^\alpha c_{\nu a}^\alpha + \sum_{i_\beta a_\beta} U_{a_\beta i_\beta}^{(0);L} c_{\mu i}^\beta c_{\nu a}^\beta \tag{5.136}$$

$$\frac{\partial P_{\mu\nu}^{(+1)}}{\partial x_L^{(-1)}} = \sum_{i_\alpha a_\beta} U_{a_\beta i_\alpha}^{(-1);L} c_{\mu i}^\alpha c_{\nu a}^\beta - \sum_{i_\beta a_\alpha} U_{a_\alpha i_\beta}^{(-1);L} c_{\mu a}^\alpha c_{\nu i}^\beta \tag{5.137}$$

$$\frac{\partial P_{\mu\nu}^{(-1)}}{\partial x_L^{(+1)}} = -\sum_{i_\alpha a_\beta} U_{a_\beta i_\alpha}^{(+1);L} c_{\mu a}^\beta c_{\nu i}^\alpha + \sum_{i_\beta a_\alpha} U_{a_\alpha i_\beta}^{(+1);L} c_{\mu i}^\beta c_{\nu a}^\alpha \tag{5.138}$$

The special form of the coupled perturbed equations are implemented in ORCArun as follows: The perturbation is of the general form $h^K \hat{s}_m$. The equations (5.133)-(5.138) and (5.139)-(5.144) below have been written in such a way that the spin integration has been performed but that the spin-dependent factors have been dropped from the right-hand sides and included in the prefactors of eqs. (5.133)-(5.135). The explicit forms of the linear equations to be solved are:

$m = 0$:

$$\left(\varepsilon_{a_\alpha}^{(0)} - \varepsilon_{i_\alpha}^{(0)}\right) U_{a_\alpha i_\alpha}^{K(0)} + a_{\text{X}} \sum_{j_\alpha b_\alpha} U_{b_\alpha j_\alpha}^{K(m)} \left\{ (b_\alpha i_\alpha | a_\alpha j_\alpha) - (j_\alpha i_\alpha | a_\alpha b_\alpha) \right\} = -\left(a_\alpha | h^K | i_\alpha\right) \tag{5.139}$$

$$\left(\varepsilon_{a_\beta}^{(0)} - \varepsilon_{i_\beta}^{(0)}\right) U_{a_\beta i_\beta}^{K(0)} + a_{\text{X}} \sum_{j_\beta b_\beta} U_{b_\beta j_\beta}^{K(m)} \left\{ (b_\beta i_\beta | a_\beta j_\beta) - (j_\beta i_\beta | a_\beta b_\beta) \right\} = -\left(a_\beta | h^K | i_\beta\right) \tag{5.140}$$

$m = +1$:

$$\left(\varepsilon_{a_\alpha}^{(0)} - \varepsilon_{i_\beta}^{(0)}\right) U_{a_\alpha i_\beta}^{K(+1)} + a_{\mathrm{X}} \sum_{j_\alpha b_\alpha} U_{b_\beta j_\alpha}^{K(+1)} \left(b_\beta i_\beta | a_\alpha j_\alpha\right) - a_{\mathrm{X}} \sum_{b_\alpha j_\beta} U_{b_\beta j_\alpha}^{K(+1)} \left(j_\beta i_\beta | a_\alpha b_\alpha\right) = -\left(a_\alpha | h^K | i_\beta\right) \tag{5.141}$$

$$\left(\varepsilon_{a_\beta}^{(0)} - \varepsilon_{i_\alpha}^{(0)}\right) U_{a_\beta i_\alpha}^{K(+1)} + a_{\mathrm{X}} \sum_{j_\beta b_\alpha} U_{b_\alpha j_\beta}^{K(+1)} \left(b_\alpha i_\alpha | a_b j_\beta\right) - a_{\mathrm{X}} \sum_{b_\beta j_\alpha} U_{b_\beta j_\alpha}^{K(+1)} \left(j_\alpha i_\alpha | a_\beta b_\beta\right) = 0 \tag{5.142}$$

$m = -1$:

$$\left(\varepsilon_{a_\beta}^{(0)} - \varepsilon_{i_\alpha}^{(0)}\right) U_{a_\beta i_\alpha}^{K(-1)} + a_{\mathrm{X}} \sum_{j_\beta b_\alpha} U_{b_\alpha j_\beta}^{K(-1)} \left(b_\alpha i_\alpha | a_b j_\beta\right) - a_{\mathrm{X}} \sum_{b_\beta j_\alpha} U_{b_\beta j_\alpha}^{K(-1)} \left(j_\alpha i_\alpha | a_\beta b_\beta\right) = -\left(a_\beta | h^K | i_\alpha\right) \tag{5.143}$$

$$\left(\varepsilon_{a_\alpha}^{(0)} - \varepsilon_{i_\beta}^{(0)}\right) U_{a_\alpha i_\beta}^{K(-1)} + a_{\mathrm{X}} \sum_{j_\alpha b_\alpha} U_{b_\beta j_\alpha}^{K(-1)} \left(b_\beta i_\beta | a_\alpha j_\alpha\right) - a_{\mathrm{X}} \sum_{b_\alpha j_\beta} U_{b_\beta j_\alpha}^{K(-1)} \left(j_\beta i_\beta | a_\alpha b_\alpha\right) = 0 \tag{5.144}$$

Note that these coupled-perturbed (CP) equations contain no contribution from the Coulomb potential or any other local potential such as the exchange-correlation potential in DFT. Hence, in the absence of HF exchange, the equations are trivially solved:

$$U_{a_\alpha i_\alpha}^{K(0)} = -\frac{\left(a_\alpha | h^K | i_\alpha\right)}{\varepsilon_{a_\alpha}^{(0)} - \varepsilon_{i_\alpha}^{(0)}} \tag{5.145}$$

$$U_{a_\beta i_\beta}^{K(0)} = -\frac{\left(a_\beta | h^K | i_\beta\right)}{\varepsilon_{a_\beta}^{(0)} - \varepsilon_{i_\beta}^{(0)}} \tag{5.146}$$

$$U_{a_\alpha i_\beta}^{K(+1)} = -\frac{\left(a_\alpha | h^K | i_\beta\right)}{\varepsilon_{a_\alpha}^{(0)} - \varepsilon_{i_\beta}^{(0)}} \tag{5.147}$$

$$U_{a_\beta i_\alpha}^{K(+1)} = 0 \tag{5.148}$$

$$U_{a_\beta i_\alpha}^{K(-1)} = -\frac{\left(a_\beta | h^K | i_\alpha\right)}{\varepsilon_{a_\beta}^{(0)} - \varepsilon_{i_\alpha}^{(0)}} \tag{5.149}$$

$$U_{a_\alpha i_\beta}^{K(-1)} = 0 \tag{5.150}$$

It is interesting that the "reverse spin flip coefficients" $U_{a_\beta i_\alpha}^{K(+1)}$ and $U_{a_\alpha i_\beta}^{K(-1)}$ are only nonzero in the presence of HF exchange. In a perturbation expansion of the CP equations they arise at second order ($\mathrm{V}^2/\Delta\varepsilon^2$) while the other coefficients are of first order ($\mathrm{V}/\Delta\varepsilon$; V represents the matrix elements of the perturbation). Hence, these contributions are of the order of α^4 and one could conceive dropping them from the treatment in order to stay consistently at the level of α^2. These terms were nevertheless kept in the present treatment.

Equations (5.139)-(5.144) are referred to as CP-SOC (coupled-perturbed spin-orbit coupling) equations. They can be solved by standard techniques and represent the desired analogue of the CP-SCF magnetic response equations solved for the determination of the g-tensor and discussed in detail earlier [710]. It is readily confirmed that in the absence of HF exchange, eqs. (5.145)-(5.150) inserted into eqs. (5.133)-(5.138) lead back to a modified Pederson-Khanna type treatment of the SOC contributions to the **D**-tensor [206]. In the framework of the formalism developed above, the Pederson-Khanna formula can be re-written in the form:

$$\begin{aligned} D_{KL}^{(\mathrm{SOC;PK})} = & \frac{1}{4S^2} \sum_{i_\beta, a_\beta} \left(\psi_i^\beta \left| h^{K;\mathrm{SOC}} \right| \psi_a^\beta\right) U_{a_\beta i_\beta}^{L;(0)} + \frac{1}{4S^2} \sum_{i_\alpha, a_\alpha} \left(\psi_i^\alpha \left| h^{K;\mathrm{SOC}} \right| \psi_a^\alpha\right) U_{a_\alpha i_\alpha}^{L;(0)} \\ & - \frac{1}{4S^2} \sum_{i_\alpha, a_\beta} \left(\psi_i^\alpha \left| h^{K;\mathrm{SOC}} \right| \psi_a^\beta\right) U_{a_\beta i_\alpha}^{L;(-1)} - \frac{1}{4S^2} \sum_{i_\beta, a_\alpha} \left(\psi_i^\alpha \left| h^{K;\mathrm{SOC}} \right| \psi_a^\alpha\right) U_{a_\alpha i_\beta}^{L;(+1)} \end{aligned} \tag{5.151}$$

This equation was derived from second-order non-self-consistent perturbation theory without recourse to spin-coupling. For the special case of no Hartree-Fock exchange, the main difference to the treatment presented here is that the correct prefactors from eqs. (5.122)-(5.124) occur in front of the spin-flip contributions rather than $\pm$ $1/(4S^2)$ in eq. (5.151). In the presence of HF exchange it is suggested that the consistent generalization of the PK method are eqs. (5.133)-(5.135) with the $\pm$ $1/(4S^2)$ prefactors and this way the method has been implemented as an option into the ORCA program.

For completeness, the evaluation of the spin-spin term in the SCF case proceeds conveniently through:

$$D_{KL}^{(\mathrm{SS})} = \frac{g_e^2}{4} \frac{\alpha^2}{S\left(2S-1\right)} \sum_{\mu\nu} \sum_{\kappa\tau} \left\{ P_{\mu\nu}^{\alpha-\beta} P_{\kappa\tau}^{\alpha-\beta} - P_{\mu\kappa}^{\alpha-\beta} P_{\nu\tau}^{\alpha-\beta} \right\} \left\langle \mu\nu \left| r_{12}^{-5} \left\{ 3 r_{12,K} r_{12,L} - \delta_{KL} r_{12}^2 \right\} \right| \kappa\tau \right\rangle \tag{5.152}$$

as derived by McWeeny and Mizuno and discussed in some detail by Sinnecker and Neese.[708] In this reference it was found that DFT methods tend to overestimate the spin-spin contribution if the calculations are based on a spin-unrestricted SCF treatment. A much better correlation with experiment was found for open-shell spin restricted calculations. The origin of this effect proved to be difficult to understand but it was shown in ref [207] that in the case of small spin-contamination, the results of ROKS calculations and of those that are obtained on the basis of the spin-unrestricted natural orbital (UNO) determinant are virtually indistinguishable. It is therefore optionally possible in the ORCA program to calculate the spin-spin term on the basis of the UNO determinant.

ZFS Keywords

Table 5.16: `%eprnmr` block input keywords relevant for ZFS calculations

Keyword	Options	Description
`dtensor`	`ss`	Calculate spin-spin part
	`so`	Calculate spin-orbit part
	`ssandso`	Calculate both parts
`DSOC`	`qro`	quasi-restricted method for SOC, must be combined with `!UNO`
	`pk`	Pederson-Khanna method for SOC
	`cp`	coupled-perturbed method for SOC (default)
	`cvw`	van Wüllen method for SOC
`DSS`	`direct`	directly use the canonical orbitals for the spin density
	`uno`	use spin density from UNOs
`printEuler`	`true/false`	Whether to calculate Euler angles via `orca_euler`

5.25 MP2 Level Magnetic Properties

Presently, hyperfine couplings (excluding the A_{orb} term), g-tensors, and shielding tensors without GIAOs can be calculated for both canonical and RI-MP2 and double-hybrid DFT without the frozen core approximation. The A_{orb} term of the hyperfine couplings is available only for RI-MP2 and double-hybrid DFT with and without frozen core approximation. In case the RIJCOSX approximation is used, the keywords `Z_GridX`, `Z_GridX_RHS`, `KCOpt`, `KC_GridX` and `KC_IntAccX` are relevant – see sections *RIJCOSX-RI-MP2 Gradients* and *MP2 and RI-MP2 Second Derivatives*. NMR shielding and g-tensor calculations with GIAOs are available for RI-MP2 and double-hybrid DFT with or without a frozen core. The implementation is described in detail in refs [347, 348] and the available options are shown in section *RI-MP2 and Double-Hybrid DFT Response Properties*. Note that for double-hybrid DFT the correct properties are printed after the density heading containing "Method : MP2" and "Level : Relaxed density". DLPNO-MP2 (and double-hybrid) shielding tensors are also available - see section *Local MP2 Second Derivatives and Response Properties*.

5.25.1 Calculating Properties from Existing Densities

Occasionally, one may calculate a density matrix using an expensive correlated method such as CCSD and realize afterwards that a certain property such as the quadrupole moment or a hyperfine coupling constant (HFCC) is also required. Rather than start the whole calculation from scratch, one may wish to use the existing density matrix to calculate the properties. For this purpose, we have experimentally introduced a "properties only" calculation mode, whereby the MOs are read from an existing *BaseName.gbw* file and the densities are read from an existing *BaseName.densities* file and only the property calculations are performed. Note however, that this presents many possibilities for error, so **only use it as a last resort and be very careful with the results**!

Take, for example, this CCSD calculation:

```
! CCSD def2-SVP

%base "BO-CCSD"

%mdci density unrelaxed end

*xyz 0 2
B 0 0 0
O 0 0 1.2049
*
```

This produces the files *BO-CCSD.gbw* and *BO-CCSD.densities*. To obtain the CCSD quadrupole moment and HFCCs without repeating the whole calculation, we can copy these two files into a new directory (highly recommended!) and start a second job with the `!PropertiesOnly` keyword. Note that the basename of the second job must be identical!

```
! CCSD def2-SVP

%base "BO-CCSD"

%mdci density unrelaxed end

*xyz 0 2
B 0 0 0
O 0 0 1.2049
*
# Everything above must be the same as in the first job!

# Request the property calculations
! PropertiesOnly
%elprop
  quadrupole true
end
%eprnmr
  Nuclei = all B { Aiso, Adip }
  Nuclei = all O { Aiso, Adip }
end
```

5.26 CASSCF Linear Response

5.26.1 Introduction

Similar to the SCF linear response (see *CP-SCF Options*), second-derivative properties can be calculated at the CASSCF level by solving the coupled perturbed (CP-)CASSCF equations for the linear response of the wavefunction parameters to a perturbation. These linear response equations are expressed as

$$\frac{\partial^2 E}{\partial \boldsymbol{\lambda}^2}\frac{\partial \boldsymbol{\lambda}}{\partial \mathbf{R}} = -\frac{\partial^2 E}{\partial \boldsymbol{\lambda} \partial \mathbf{R}}$$

where $\boldsymbol{\lambda}$ are the CASSCF wavefunction parameters and $\mathbf{R}$ are the perturbations for which the response equations are being solved [711, 712]. The property gradient on the right-hand side (RHS) is a known perturbation-dependent quantity, but the left-hand side (LHS) depends on the solution of the response of the wavefunction parameters to $\mathbf{R}$. Therefore, the response to the perturbation must be solved for iteratively, which is done using the trial vectors $\mathbf{X}$. This leads to the LHS being computed as a sigma-vector, which is reassembled at every iteration. The linear response equation for perturbation R_i can then be written as

$$\boldsymbol{\sigma} = \mathbf{H}\mathbf{X}^{(R_i)} = -\mathbf{G}^{(R_i)}$$

where $\mathbf{H}$ is the perturbation-independent Hessian matrix and $\mathbf{G}$ is the perturbation-specific RHS matrix. Once the linear equation converges, the solution vector of the response parameters can then be contracted with the electron- and spin-density matrices to yield the AO response density matrices, $\left[\mathbf{P}^{\alpha\pm\beta}\right]^{(\mathbf{R})}$. The response densities can then be contracted with appropriate AO 1-electron property integrals to compute the second-order contributions to second-derivative properties.

In ORCA these equations are solved by the `orca_casscfresp` program and the underlying solver is `BHP22`, a Davidson-type linear equation solver. The RHS is built from the property integrals calculated in the `orca_propint` program. After the solution converges and the response densities are made and stored, the `orca_prop` program is called, wherein the appropriate densities and response densities are contracted with the necessary property integrals (also from `orca_propint`). This use of densities keeps the response property calculations in `orca_prop` generally applicable to all methods as the densities house the method-specific information.

5.26.2 Technical Notes

It should be noted that CASSCF linear response uses the optimized CASSCF wavefunction as a starting point. Thus, a `%casscf` block with the appropriate inputs (see *CASSCF*) must be provided in the input. State-specific (SS-)CASSCF response is run on SS (`NRoots 1`) CASSCF wavefunctions. If the CASSCF wavefunction is state-averaged (SA), the response is run over the averaged manifold of states and not on a specific root. In this case, one should analyze the output carefully. It is wise to only average states which are (nearly-)degenerate.

By default, a CASSCF calculation will be run from scratch before running the CASSCF Response. Alternatively, orbitals from a previously converged CASSCF calculation may be used with `!MOREAD NoIter` and supplying the appropriate `.gbw` file via `%moinp`. In this case, be sure that the input in the `%casscf` block matches that from the same input block of the previously converged calculation and be sure to check the orbitals well!

The linear response equations are solved iteratively until the convergence threshold for the residual norm (`TolR`) is reached. This value can be set by the user, but ORCA will set it by default to the tightest reasonable convergence criteria, taking into account the orbital- and CI-gradient tolerances of the underlying CASSCF calculation. This should be the most reasonable setting; be careful if trying to tighten this threshold!

The appropriate property flags in the `%elprop` and `%eprnmr` blocks must be set to calculate the properties that are wanted (see *Electrical Properties - Electric Moments and Polarizabilities* and *EPRNMR - keywords for magnetic properties*). While all first-derivative properties work with CASSCF, not all second-derivative properties are currently available with CASSCF. The static properties currently available (sorted by perturbation taken for the response) include:

- Electric ($\mathbf{E}$) Field
  - Dipole/dipole Polarizability

  - Dipole/quadrupole Polarizability
- Quadrupole (**Q**) Field
  - Quadrupole/Quadrupole Polarizability
- Velocity (**v**)
  - Velocity Polarizability
- Magnetic (**B**) Field (**without** GIAOs)
  - EPR g-Tensor
  - NMR Chemical Shieldings
- Nuclear Magnetic Moment ($\mathbf{I}^{(A)}$)
  - EPR A-matrix (Aorb contribution)

The magnetic properties are currently implemented without gauge-including atomic orbitals (GIAOs). Thus, an appropriate gauge-origin must be provided in the `%eprnmr` block! For the EPR g-tensor, using a large basis set with a chemically relevant gauge origin is recommended. However, one should in general be wary of NMR chemical shieldings without GIAOs.

5.26.3 Basic Usage

The following is an overview of the blocks required for a property calculation with

```
! def2-TZVP AutoAux    # (or other appropriate basis & auxiliary)

%casscf
  ...
end

%casresp     # only required to change solver, printing, or preconditioner options
  ...
end

%elprop      # only required if electric property requested
  ...
end

%eprnmr      # only required if magnetic property requested
  ...
  ORI ...
end

* xyzfile ...
```

5.26.4 Troubleshooting

If the `%casresp` block is specified and the calculation does not run through the CASSCF Response section, then the properties requested are not second-derivatives and therefore do not require linear response equations to be solved. If, however, the job aborts, watch for these possibilities:

- At least one of the second-derivative properties requested is not available at the CASSCF level (see list above of those currently implemented)
- `MaxIter` may need to be increased or `TolR` loosened
- If one or more perturbations meet convergence criteria and then later do not, try turning off solution locking.
  - This may point to `TolR` being too tight

- A magnetic field property is requested without setting the gauge-origin in the `%eprnmr` block
- An appropriate auxiliary basis (or the `!AutoAux` keyword) may be required, especially for:
  - The trial vector preconditioner (which is on by default)
  - `TrafoStep RI` in the `%casscf` block
  - RIJCOSX, RIJONX
- RIJK was specified (this is **NOT** supported)

Note that the solver-related options in the `%elprop` and `%eprnmr` blocks do not affect the solution of the CP-CASSCF equations.

5.26.5 Notes on Printing

The information on the types of property integrals computed can be found in the `ORCA PROPERTY INTEGRAL CALCULATIONS` section of the output file. The `orca_casscfresp` output begins with the header `ORCA CASSCF RESPONSE CALCULATION`. After information about the types and number of perturbations, the calculation splits into the major types of perturbations: real and imaginary. All response equations of the same type can be solved simultaneously.

Each of these types has its own section in the output file. These sections begin with information on the orbital ranges and the CI space, then go into the iterative solution of the equations. The printout here gives an overview of the iteration number, the residual norm of each response equation, and if that response equation has met the convergence criteria. The following is an example of this output (with iterations 3–7 removed for simplicity):

```
 ------------------------------------------------------------------------------
----------

                      SOLVER Davidson-type linear equation solver

   Iter.                  ||Error||_2   Conv.                          (TolR = 1.
000e-08)
 ------------------------------------------------------------------------------
----------
    0  (rhs    0)        1.645417e-01     No
       (rhs    1)        1.826041e-01     No
       (rhs    2)        2.047543e-01     No
    1  (rhs    0)        7.069575e-02     No
       (rhs    1)        3.363295e-02     No
       (rhs    2)        7.229274e-02     No
    2  (rhs    0)        9.183136e-03     No
       (rhs    1)        3.922435e-03     No
       (rhs    2)        7.152764e-03     No
(...)
 ------------------------------------------------------------------------------
----------

     All 3 Linear Equations CONVERGED Below 1.000e-08

 ------------------------------------------------------------------------------
----------
    8  (rhs    0)        4.070240e-09    Yes
       (rhs    1)        1.186846e-09    Yes
       (rhs    2)        4.407666e-09    Yes
```

Before going on to the necessary property modules, the significant contributions to the RHS and/or response vectors will be outputted if they were requested (via `PrintRHSVec` and `PrintRspVec`). Here, outputs such as the following can be seen.

```
  ---------------------------------
   CASSCF Response Vector Analysis
  ---------------------------------
    square of coefficients of particle-hole and CAS-CI excitations are printed if␣
↪larger than 1.0e-02


IPERT: 0

   (-) I   9 (   9) -> V   0 (  17) :      0.07167 (c= -0.26772137)
(...)
   (-) A   1 (  14) -> V  13 (  30) :      0.07323 (c=  0.27061481)
(...)
   (-) I   4 (   4) -> A   0 (  13) :      0.02915 (c=  0.17074032)
(...)
   (-) CI                                  1.65592 [     7]: 1201
(...)
```

The majority of the output has been removed for the sake of simplicity. Here, the significant contributions to the response vector are printed if the square of the vector element is larger than `1.0e-02` (the default value for `TolPrintVec`). Each line begins with either a `(+)` or `(-)`, which denote the Hermiticity of the excitation operator. With static CASSCF linear response, `(+)` always denotes an imaginary perturbation and `(-)` a real perturbation. For the first vector (i.e. `IPERT: 0`), four contributions have been listed here.

The first three are from particle-hole excitations going from the left size of the arrow to the right. On each side of the arrow has three things: a letter designating if it is an inactive (`I`), active (`A`), or virtual (`V`) orbital; a number designating the index within that orbital subblock, and a number in parentheses designating the overall orbital number. Further to the right, the coefficient (i.e. the vector element) is given in parentheses and the square of that value is given to the left of it. This is how the relative contributions of each excitation can be analyzed.

The final contribution shown is from a CI excitation. Its weight is listed, followed by the index in brackets and the corresponding configuration of the active electrons among the active orbitals.

5.26.6 Keywords

Table 5.17: `%casresp` block input keywords.

Keyword	Options	Description
TolR	<real>	Convergence threshold (residual norm of LR equations) *(default) uses tightest reasonable threshold*
MaxIter	64	Maximum number of iterations
MaxRed	100	Maximum size of the reduced space per RHS
PrintRHSVec	false	Print significant contributions to the RHS vector
PrintRspVec	false	Print significant contributions to the response vector
PrintWF	CFG	*(default)* Print CI part of RHS/rsp vector in configuration basis
	CSF	Print CI part of RHS/rsp vector in CSF basis
	DET	Print CI part of RHS/rsp vector in determinant basis
TolPrintVec	1.e-2	Threshold to print a contribution to the RHS/response vector
PreCondType	full	*(default)* Full preconditioner of trial vector (requires aux basis)
	diag	Diagonal preconditioner of trial vector (requires aux basis)
	none	No preconditioning of trial vector
PreCondMaxRed	400	Hessian precondition subset size
DoOrbResp	true	Include orbital response
DoLocking	true	Lock solutions to perturbations as they meet convergence criteria
DoOlsen	false	Use Olsen preconditioner

5.27 Mössbauer Parameters

^{57}Fe Mössbauer spectroscopy probes the transitions of the nucleus between the $I = \frac{1}{2}$ ground state and the $I = \frac{3}{2}$ excited state at 14.4 keV above the ground state. The important features of the Mössbauer spectrum are the isomer shift (δ) and the quadrupole splitting (ΔE_{Q}). An idealized spectrum is shown in Fig. 5.64.

Fig. 5.64: An idealized Mössbauer spectrum showing both the isomer shift, δ, and the quadrupole splitting, ΔE_{Q}.

5.27.1 Isomer Shift

The isomer shift measures the shift in the energy of the γ-ray absorption relative to a standard, usually Fe foil. The isomer shift is sensitive to the electron density at the nucleus, and indirectly probes changes in the bonding of the valence orbitals due to variations in covalency and 3d shielding. Thus, it can be used to probe oxidation and spin states, and the coordination environment of the iron.

Both the isomer shift and quadrupole splitting can be successfully predicted using DFT methods. The isomer shift is directly related to the s electron density at the nucleus and can be calculated using the formula

$$\delta = \alpha(\rho_0 - C) + \beta \tag{5.153}$$

where α is a constant that depends on the change in the distribution of the nuclear charge upon absorption, and ρ_0 is the electron density at the nucleus [713]. The constants α and β are usually determined via linear regression analysis of the experimental isomer shifts versus the theoretically calculated electron density for a series of iron compounds with various oxidation and spin states. Since the electron density depends on the functional and basis set employed, fitting must be carried out for each combination used. A compilation of calibration constants (α, β and C) for various methods was assembled.[714] Usually an accuracy of better than 0.10 mm s^{-1} can be achieved for DFT with reasonably sized basis sets.

5.27.2 Quadrupole splitting

The quadrupole splitting arises from the interaction of the nuclear quadrupole moment of the excited state with the electric field gradient at the nucleus. The former is related to the non-spherical charge distribution in the excited state. As such it is extremely sensitive to the coordination environment and the geometry of the complex.

The quadrupole splitting is proportional to the largest component of the **electric field gradient** (EFG) tensor at the nucleus and can be calculated using the formula:

$$\Delta E_{\mathrm{Q}} = \frac{1}{2} e Q V_{zz} \left(1 + \frac{\eta^2}{3}\right)^{\frac{1}{2}} \tag{5.154}$$

where e is the electrical charge of an electron and Q is the nuclear quadrupole moment of Fe (approximately 0.16 barns). V_{xx}, V_{yy} and V_{zz} are the electric field gradient tensors and η, defined as

$$\eta = \left| \frac{V_{xx} - V_{yy}}{V_{zz}} \right| \tag{5.155}$$

is the asymmetry parameter in a coordinate system chosen such that $|V_{zz}| \geq |V_{yy}| \geq |V_{xx}|$.

The electric field gradient tensor is closely related to the dipole contribution to the hyperfine coupling. The main differences are that the electron instead of the spin density enters its calculation and that it contains a nuclear contribution due to the surrounding nuclei. It is calculated from

$$\begin{aligned} V_{\mu\nu}(N) = &- \sum_{kl} P_{kl} \left\langle \phi_k \left| r_N^{-5} \left(3\vec{r}_{N\mu}\vec{r}_{N\nu} - \delta_{\mu\nu} r_N^2\right) \right| \phi_l \right\rangle \\ &+ \sum_{A \neq N} Z_A \vec{R}_{AN}^{-5} \left(3\vec{R}_{AN\mu}\vec{R}_{AN\nu} - \delta_{\mu\nu} R_{AN}^2\right) \end{aligned} \tag{5.156}$$

with Z_A as the nuclear charge of nucleus A and $\vec{R}_{AN}$ as a vector of magnitude R_{AN} that points from nucleus A to nucleus N. **P** is the first order density matrix.

5.27.3 Basic usage

An example of how to calculate the electron density and quadrupole splitting of an iron center is as follows:

```
%eprnmr
  nuclei = all Fe {fgrad, rho}
end
```

If a basis set with tight functions such as the CP(PPP) is employed, one might have to increase the radial integration accuracy for the iron atom. From ORCA 5.0, this is considered during grid construction and the defaults should work very well. However for very problematic cases it can be increased by controlling the `SPECIALGRIDINTACC` flag under `%METHOD` (see Sec. *The Radial Grid Scheme* for details).

The output file should contain the following lines, where you obtain the calculated quadrupole splitting directly and the RHO(0) value (the electron density at the iron nucleus). To obtain the isomer shift one has to insert the RHO(0) value into the appropriate linear equation (Eq. (5.153)).

```
Moessbauer quadrupole splitting parameter (proper coordinate system)
e**2qQ  = -0.406 MHz = -0.035 mm/s
eta     = 0.871
Delta-EQ=(1/2{e**2qQ}*sqrt(1+1/3*eta**2) = -0.227 MHz = -0.020 mm/s
RHO(0)=  11581.352209571 a.u.**-3   # the electron density at the Fe nucleus.
```

Note

- Following the same procedure, Mössbauer parameters can be computed with the CASSCF wavefunction. In case of a state-averaged CASSCF calculation, the averaged density is used in the subsequent Mössbauer calculation.

5.27.4 Keywords

All properties tied to a given nucleus in the molecule (NMR shielding, hyperfine coupling, quadrupolar coupling) share a common input syntax. This syntax consists of a selection (the nuclei for which you want to do something) and a list of actions you want the program to take for these nuclei inside curly brackets. A complete list of possible keywords for the eprnmr module can be found in *EPRNMR - keywords for magnetic properties*. It's best explained on an example. The following

```
%eprnmr
  nuclei = all O { aiso, adip, fgrad, shift, ist=17 }
end
```

would select all oxygen atoms in the molecule, pick the ^{17}O isotope, and request isotropic + spin-dipolar hyperfine couplings, the electric field gradient, and NMR shielding to be calculated for them.

Warning

All nuclei specified on one line will be assigned with the same isotopic mass (and other parameters)!

Table 5.18: `%eprnmr` block input keywords relevant for Mössbauer calculations

Keyword	Selection	What to do	Description
`nuclei`	`all <type>`		Selects all nuclei of element `<type>`
	`i1,i2,i3...`		Selects atoms `i1`, `i2`, `i3`... (starts at 1!)
		`{ rho }`	Calculates density at nucleus for selection
		`{ fgrad }`	Calculates EFG for selection
		`{ ist=<N> }`	Selects isotope `<N>` for selection
		`{ QQQ=<N> }`	Sets quadrupole moment to `<N>` for selection
		`{ III=<N> }`	Sets nuclear spin to `<N>` for selection
		`{ fgrad, rho, ist=57 }`	Example requesting multiple actions for selection

5.28 The Spin-Orbit Coupling Operator

Several variants of spin-orbit coupling operators can be used for property calculations [715]. They are based on effective potential and mean-field approaches, and have various parameters that can be selected via the `%rel` block. Note that the SOMF operator depends on the density matrix, so the operator itself can differ for example between a CASSCF and an MRCI calculation.

Note: The defaults have slightly changed in ORCA 5.0, see `SOCFlags` in the following.

```
%rel
     # ---------------------------------------------------
     # SPIN ORBIT COUPLING OPERATORS
     # ---------------------------------------------------
     SOCType 0  # none
```

(continues on next page)

(continued from previous page)

```
            1  # effective nuclear charge
            2  # DEPRECATED
            3  # mean-field/effective potential (default)
            4  # DEPRECATED
    # ---------------------------------------------------
    # Flags for construction of potential; operative
    # only for SOCType 3.
    # ---------------------------------------------------
    SOCFlags 1,4,3,0 # default if nothing is specified
                     # 1,3,3,0 default if RI is applied and AuxJ available
                     # e.g. when using !RIJCOSX (default for DFT) or !RIJONX
    # Flag 1 = 0 - do not include 1-electron terms
    #        = 1 - do include 1-electron terms
    # Flag 2 = 0 - do not include Coulomb terms
    #        = 1 - compute Coulomb terms fully numeric
    #        = 2 - compute Coulomb term seminumeric
    #        = 3 - compute Coulomb term with RI approx
    #        = 4 - compute Coulomb term exactly
    # Flag 3 = 0 - do not include exchange terms
    #        = 1 - do include local X-alpha exchange
    #              the X-Alpha parameter can be chosen via
    #              % rel Xalpha 0.7 (default)
    #        = 2 - same as 1 but with sign reversed
    #        = 3 - exchange via one-center exact
    #              integrals including the spin-other
    #              orbit interaction
    #        = 4 - all exchange terms full analytic
    #              (this is expensive)
    # Flag 4 = 0 - do not include DFT local correlation
    #              terms
    #        = 1 - do include local DFT correlation (here
    #              this is done with VWN5)
    #
    SOCMaxCenter 4 # max. number of centers to include in
                   # the integrals (not fully consistently
                   # implemented yet; better leave equal to 4)
    #  The following simple input equivalents can also be used:
    #  SOMF(1X)      = SOCType 3, SOCFlags 1,2,3,0 and SOCMaxCenter 4
    #  RI-SOMF(1X)   = SOCType 3, SOCFlags 1,3,3,0 and SOCMaxCenter 4
    #  VEFF-SOC      = SOCType 3, SOCFlags 1,3,1,1 and SOCMaxCenter 4
    #  VEFF(-2X)-SOC = SOCType 3, SOCFlags 1,3,2,1 and SOCMaxCenter 4
    #  AMFI          = SOCType 3, SOCFlags 1,4,3,0 and SOCMaxCenter 1
    #  NOTE: If you choose the RI option you need to specify an auxiliary basis␣
↪set
    #        even if the underlying SCF calculation does not make use of any form
    #        of RI!
    # ------------------------------------------------
    # For the effective nuclear charge SOC operator
    # the nuclear charges can be adjusted.
    # ------------------------------------------------
    Zeff[26] 0.0   # set the effective nuclear charge
                   # of iron (Z = 26) to zero
    # ------------------------------------------------
    # Neglecting SOC contributions from particular
    # atoms
    # ------------------------------------------------
    SOCOff 0,5     # turn off the SOC for atoms 0 and 5
                   # this makes sense if the SOC operator
                   # has only one center contributions
                   # (e.g. effective nuclear charge)
```

Simple input equivalents are listed in Table 5.19 and described in more detail in [715]. More details on the (depre-

cated) AMFI-A approach which uses pre-calculated atomic densities can be found in [716].

The Breit-Pauli spin-orbit coupling operator is given by:

$$\hat{H}_{\text{SOC}} = \hat{H}^{(1)}_{\text{SOC}} + \hat{H}^{(2)}_{\text{SOC}}$$

with the one- and two-electron contributions

$$\hat{H}^{(1)}_{\text{SOC}} = \frac{\alpha^2}{2} \sum_i \sum_A Z_A \frac{(\mathbf{r}_i - \mathbf{R}_A) \times \mathbf{p}_i}{|\mathbf{r}_i - \mathbf{R}_A|^3} \hat{s}_i \equiv \frac{\alpha^2}{2} \sum_i \sum_A Z_A r_{iA}^{-3} \hat{\mathbf{l}}_{iA} \hat{\mathbf{s}}_i \tag{5.157}$$

$$\hat{H}^{(2)}_{\text{SOC}} = -\frac{\alpha^2}{2} \sum_i \sum_{j \neq i} \frac{(\mathbf{r}_i - \mathbf{r}_j) \times \mathbf{p}_i}{|\mathbf{r}_i - \mathbf{r}_j|^3} (\hat{\mathbf{s}}_i + 2\hat{\mathbf{s}}_j) \tag{5.158}$$

$$\equiv -\frac{\alpha^2}{2} \sum_i \sum_{j \neq i} \hat{\mathbf{l}}_{ij} r_{ij}^{-3} (\hat{\mathbf{s}}_i + 2\hat{\mathbf{s}}_j) \tag{5.159}$$

Note

One attractive possibility is to represent the SOC by the spin-orbit mean-field (SOMF) method developed by Hess et al.,[717] widely used in the AMFI program by Schimmelpfennig [718] and discussed in detail by Berning et al.[719] as well as in ref. [715]. In terms of an (orthonormal) one-electron basis, the matrix elements of the SOMF operator are:

$$\begin{aligned} h_{rs}^{K;\text{SOC}} &= \left(p \left| \hat{h}_K^{1el-\text{SOC}} \right| q\right) \\ &+ \sum_{rs} P_{rs} \left[\left(pq \left| \hat{g}_K^{\text{SOC}} \right| rs\right) - \tfrac{3}{2} \left(pr \left| \hat{g}_K^{\text{SOC}} \right| sq\right) - \tfrac{3}{2} \left(sq \left| \hat{g}_K^{\text{SOC}} \right| pr\right) \right] \end{aligned} \tag{5.160}$$

and:

$$\hat{h}_k^{1el-\text{SOC}} (\mathbf{r}_i) = \frac{\alpha^2}{2} \sum_i \sum_A Z_A r_{iA}^{-3} \hat{\mathbf{l}}_{iA;k} \tag{5.161}$$

$$\hat{g}_k^{\text{SOC}} (\mathbf{r}_i, \mathbf{r}_j) = -\frac{\alpha^2}{2} \hat{\mathbf{l}}_{ij;k} r_{ij}^{-3} \tag{5.162}$$

$\hat{\mathbf{l}}_{iA} = (\hat{\mathbf{r}}_i - \mathbf{R}_A) \times \hat{\mathbf{p}}_i$ is the angular momentum of the i'th electron relative to nucleus A. The vector $\hat{\mathbf{r}}_{iA} = \hat{\mathbf{r}}_i - \mathbf{R}_A$ of magnitude r_{iA} is the position of the i'th electron relative to atom A. Likewise, the vector $\hat{\mathbf{r}}_{ij} = \hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j$ of magnitude r_{ij} is the position of the ith electron relative to electron j and $\hat{\mathbf{l}}_{ij} = (\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j) \times \hat{\mathbf{p}}_i$ is its angular momentum relative to this electron. $\mathbf{P}$ is the charge density matrix of the electron ground state ($P_{pq} = \left\langle 0SS \left| E_q^p \right| 0SS \right\rangle$with $E_q^p = a_{p\beta}^+ a_{q\beta} + a_{p\alpha}^+ a_{q\alpha}$ where $a_{p\sigma}^+$ and $a_{q\sigma}$ are the usual Fermion creation and annihilation operators).

This operator would be hard to handle exactly; therefore it is common to introduce mean field and/or effective potential approaches in which the operator is written as an effective one-electron operator:

$$\hat{H}_{\text{SOC}} \cong \sum_i \hat{\mathbf{h}}_i^{(\text{eff})} \hat{\mathbf{s}}_i \tag{5.163}$$

The simplest approximation is to simply use the one-electron part and regard the nuclear charges as adjustable parameters. Reducing their values from the exact nuclear charge is supposed to account in an average way for the screening of the nuclear charge by the electrons. In our code we use the effective nuclear charges of Koseki et al. ([720, 721, 722]) This approximation introduces errors which are usually smaller than 10% but sometimes are larger and may approach 20% in some cases. The approximation is best for first row main group elements and the first transition row (2p and 3d elements). For heavier elements it becomes unreliable.

A much better approximation is to take the two-electron terms into account precisely. Without going into details here – the situation is as in Hartree-Fock (or density functional) theory and one gets Coulomb, exchange and correlation terms. The correlation terms (evaluated in a local DFT fashion) are negligible and can be safely neglected. They are optionally included and are not expensive computationally. The Coulomb terms is (after the one-electron term) the

second largest contribution and is expensive to evaluate exactly. The situation is such that in the Coulomb-part the spin-other orbit interaction (the second term in the two-electron part) does not contribute and one only has to deal with the spin-own-orbit contribution. The exact evaluation is usually too expensive to evaluate. The RI and seminumeric approximation are much more efficient and introduce only minimal errors (on the order of usually not more than 1 ppm in g-tensor calculations for example) and are therefore recommended. The RI approximation is computationally more efficient. Please note that you have to specify an auxiliary basis set to take advantage of the RI approximation, even if the preceding SCF calculation does not make use of any form of RI. The one-center approximation to the Coulomb term introduces much larger errors. The fully numeric method is both slower and less accurate and is not recommended.

The exchange term has contributions from both the spin-own-orbit and spin-other-orbit interaction. These are taken both into account in the mean-field approximation which is accessed by `Flag 3 = 3`. Here a one-center approximation is much better than for the Coulomb term since both the integrals and the density matrix elements are short ranged. Together with the Coulomb term this gives a very accurate SOC operator which is recommended. The DFT-Veff operator suffers from not treating the spin-other-orbit part in the exchange which gives significant errors (also, local DFT underestimates the exchange contributions from the spin-same-orbit interaction by some 10% relative to HF but this is not a major source of error). However, it is interesting to observe that in the precise analytical evaluation of the SOMF operator, the spin-other-orbit interaction is exactly -2 times the spin-own-orbit interation. Thus, in the DFT framework one gets a *much* better SOC operator if the sign of the DFT exchange term is simply reversed! This is accessed by `Flag 3 = 2`.

5.28.1 Exclusion of Atomic Centers

In ORCA it is possible to change the spin-orbit coupling operator in order to exclude contributions from user-defined atoms. This approach can be useful, for example, in quantifying the contribution of the ligands to the zero-field splitting (ZFS); for an application of this method see Ref. [723].

This is illustrated for the calculation of the SOC contribution to the ZFS of the triplet oxygen molecule. Using the input below we start by a normal calculation of the ZFS, including both oxygen atoms. Note that we use here the effective nuclear charge operator. This is required as not all implemented SOC operators are compatible with the decomposition in terms of individual centers contributions.

```
! def2-TZVP def2-TZVP/c

%casscf nel 8
        norb 6
        mult 3,1
        nroots 1,3
        rel dosoc true
            end
        end

%rel
    SOCType 1
    end

*xyz 0 3
O 0 0 0
O 0 0 1.207
*
```

The calculated value of the D parameter is approximately 2.573 cm^{-1}. In a second calculation we exclude the contribution from the first oxygen atom. For this we change the `%rel` block to the one below.

```
%rel
    SOCType 1
    SOCOff 0
    end
```

Now the D parameter is calculated to be approximately 0.643 cm^{-1}, a result that deviates quite significantly from half of the value calculated previously, implying that non-additive effects are important. In addition to the effective nuclear

charge operator, the AMFI-A operator described previously can be used. Given that this is based on precalculated atomic densities, it might be preferred for heavier elements where the effective nuclear charge operator becomes unreliable. The method is not limited to CASSCF calculations as described above, and can be used in DFT, MRCI and ROCIS calculations.

5.28.2 Simple Input Keywords

Table 5.19: Simple input keywords to choose the SOC operator handling and their equivalent `%rel` block settings

Keyword	SOCType	SOCFlags	SOCMaxCenter
ZEFF-SOC	1		
SOMF	3	1,4,4,0	4
SOMF(1X)	3	1,2,3,0	4
RI-SOMF	3	1,3,3,0	4
RI-SOMF(1X)	3	1,3,3,0	4
SOMF(4X)	3	1,2,4,0	4
RI-SOMF(4X)	3	1,3,4,0	4
VEFF-SOC	3	1,3,1,1	4
VEFF(-2X)-SOC	3	1,3,2,1	4
AMFI	3	1,4,3,0	1

5.29 Broken-Symmetry Wavefunctions and Exchange Couplings

A popular way to estimate the phenomenological parameter J_{AB} that enters the Heisenberg–Dirac–van Vleck Hamiltonian which parameterizes the interaction between two spin systems is the "broken-symmetry" formalism. The phenomenological Hamiltonian is:

$$H_{\mathrm{HDvV}} = -2J_{\mathrm{AB}}\vec{S}_{\mathrm{A}}\vec{S}_{\mathrm{B}} \tag{5.164}$$

It is easy to show that such a Hamiltonian leads to a "ladder" of spin states from $S = S_{\mathrm{A}}+S_{\mathrm{B}}$ down to $S = |S_{\mathrm{A}} - S_{\mathrm{B}}|$. If the parameter J_{AB} is positive the systems "A" and "B" are said to be *ferromagnetically* coupled because the highest spin-state is lowest in energy while in the case that J_{AB} is negative the coupling is *antiferromagnetic* and the lowest spin state is lowest in energy.

In the broken symmetry formalism one tries to obtain a wavefunction that breaks spatial (and spin) symmetry. It may be thought of as a "poor man's MC-SCF" that simulates a multideterminantal character within a single determinant framework. Much could be said about the theoretical advantages, disadvantages, problems and assumptions that underly this approach. Here, we only want to show how this formalism is applied within ORCA.

For N_{A} unpaired electrons localized on "site A" and N_{B} unpaired electrons localized on a "site B" one can calculate the parameter J_{AB} from two separate spin-unrestricted SCF calculations: (a) the calculation for the high-spin state with $S = \frac{(N_{\mathrm{A}}+N_{\mathrm{B}})}{2}$ and (b) the "broken symmetry" calculation with $M_S = \frac{(N_{\mathrm{A}}-N_{\mathrm{B}})}{2}$ that features N_{A} spin-up electrons that are quasi-localized on "site A" and N_{B} spin-down electrons that are quasi-localized on "site B". Several formalisms exist to extract J_{AB}: [724, 725, 726, 727, 728, 729].

$$J_{\mathrm{AB}} = -\frac{(E_{\mathrm{HS}} - E_{\mathrm{BS}})}{(S_{\mathrm{A}} + S_{\mathrm{B}})^2} \tag{5.165}$$

$$J_{\mathrm{AB}} = -\frac{(E_{\mathrm{HS}} - E_{\mathrm{BS}})}{(S_{\mathrm{A}} + S_{\mathrm{B}})\,(S_{\mathrm{A}} + S_{\mathrm{B}} + 1)} \tag{5.166}$$

$$J_{\mathrm{AB}} = -\frac{(E_{\mathrm{HS}} - E_{\mathrm{BS}})}{\langle S^2\rangle_{\mathrm{HS}} - \langle S^2\rangle_{\mathrm{BS}}} \tag{5.167}$$

We prefer the last definition (Eq. (5.167)) because it is approximately valid over the whole coupling strength regime while the first equation implies the weak coupling limit and the second the strong coupling limit.

In order to apply the broken symmetry formalism use:

```
%scf BrokenSym NA,NB
     end
```

The program will then go through a number of steps. Essentially it computes an energy and wavefunction for the high-spin state, localizes the orbitals and reconverges to the broken symmetry state.

Caution

Make sure that in your input coordinates "site A" is the site that contains the larger number of unpaired electrons!

Most often the formalism is applied to spin coupling in transition metal complexes or metal-radical coupling or to the calculation of the potential energy surfaces in the case of homolytic bond cleavage. In general, hybrid DFT methods appear to give reasonable semiquantitative results for the experimentally observed splittings.

As an example consider the following simple calculation of the singlet–triplet splitting in a stretched Li_2 molecule:

```
#
# Example of a broken symmetry calculation
#
! B3LYP DEF2-SVP TightSCF
%scf BrokenSym 1,1
     end
* xyz 0 3
  Li  0  0  0
  Li  0  0  4
*
```

There is a second mechanism for generating broken-symmetry solutions in ORCA. This mechanism uses the individual spin densities and is invoked with the keywords `FlipSpin` and `FinalMs`. The strategy is to exchange the α and β spin blocks of the density on certain user-defined centers after converging the high-spin wavefunction. With this method arbitrary spin topologies should be accessible. The use is simple:

```
#
# Example of a broken symmetry calculation using the "FlipSpin" feature
#
! B3LYP DEF2-SVP TightSCF
%scf
     FlipSpin 1
       # Flip spin is a vector and you can give a list of atoms
       # on which you want to have the spin flipped. For example
       #  FlipSpin 17,38,56
       # REMEMBER: counting starts at atom 0!
     FinalMs  0
       # The desired Ms value of the broken symmetry determinant.
       # This value MUST be given since the program cannot determine it by itself.
     end
* xyz 0 3
  Li  0  0  0
  Li  0  0  4
*
```

Finally, you may attempt to break the symmetry by using the SCF stability analysis feature (see Section *SCF Stability Analysis*). The ground spin state can be obtained by diagonalizing the above spin Hamiltonian through **ORCA-ECA** utility (see *orca_eca*).

5.29.1 Approximate Spin Projection Method

The approximate spin projection (AP) method, proposed by Yamaguchi and co-workers, is a technique to remove the spin contamination from exchange coupling constants.[728, 730, 731] In this scheme, the energy of a system is given by

$$E_{\mathrm{AP}} = \alpha E_{\mathrm{BS}} - (\alpha - 1)\, E_{\mathrm{HS}} \tag{5.168}$$

The parameter α is calculated as

$$\alpha = \frac{S^{\mathrm{HS}}\left(S^{\mathrm{HS}}+1\right) - S_Z^{\mathrm{BS}}\left(S_Z^{\mathrm{BS}}+1\right)}{\langle\hat{\mathbf{S}}^2\rangle^{\mathrm{HS}} - \langle\hat{\mathbf{S}}^2\rangle^{\mathrm{BS}}} \tag{5.169}$$

Here, S_Z^{BS} is the z-component of the total spin for the BS state ($S_Z^{\mathrm{BS}} = (N_\alpha - N_\beta)/2$). Alternatively, one can adopt Noodleman's scheme,[726] where α is calculated as follows

$$\alpha = \frac{S^{\mathrm{HS}}\left(S^{\mathrm{HS}}+1\right) - S_Z^{\mathrm{BS}}\left(S_Z^{\mathrm{BS}}+1\right)}{\left(S^{\mathrm{HS}}\right)^2} \tag{5.170}$$

or Ruiz's scheme,[732] with α equal to

$$\alpha = \frac{S^{\mathrm{HS}}\left(S^{\mathrm{HS}}+1\right) - S_Z^{\mathrm{BS}}\left(S_Z^{\mathrm{BS}}+1\right)}{S^{\mathrm{HS}}\left(S^{\mathrm{HS}}+1\right)} \tag{5.171}$$

The AP method is requested via the tag `APMethod` in the `%scf` block:

```
%scf BrokenSym NA,NB
     APMethod 3   # 1 = Noodleman
                  # 2 = Ruiz
                  # 3 = Yamaguchi
end
```

The default is `APMethod 0`, which corresponds to a normal BS calculation. Yamaguchi's AP method is available for single point energy calculations and geometry optimizations. If we run a geometry optimization in the context of Yamaguchi's AP method, then, the gradient of equation (5.168) w.r.t a nuclear displacement X reads as

$$\frac{\partial E_{\mathrm{AP}}}{\partial X} = \alpha\frac{\partial E_{\mathrm{BS}}}{\partial X} - (\alpha-1)\frac{\partial E_{\mathrm{HS}}}{\partial X} + \frac{\partial \alpha}{\partial X}\left(E_{\mathrm{BS}} - E_{\mathrm{HS}}\right) \tag{5.172}$$

The last term contains the derivative $\frac{\partial\alpha}{\partial X}$. ORCA uses the formalism proposed by Saito and Thiel, which involves solving the CP-SCF equations in each geometry optimization cycle.[733] The cost of such a calculation is higher than using Noodleman's or Ruiz's schemes, where $\frac{\partial\alpha}{\partial X} = 0$.

5.30 Decomposition of the Magnetic Exchange Coupling

The Decomposition method ([734, 735, 736, 737]) aims to extract the main physical contributions to the magnetic exchange coupling J_{ab} between two magnetic sites A and B based on successive relaxation of high-spin (HS) and broken symmetry determinants. The present implementation in ORCA refers to systems involving two magnetic centres and is based on the use of the Yamaguchi formula. For systems involving more centres, the Yamaguchi formula is invalid, and the decomposition implies a different strategy for which an input file generator to perform the decomposition with ORCA is available here.([738, 739])

Based on the Heisenberg-Dirac-van Vleck Hamiltonian,

$$H_{\mathrm{HDvV}} = -2J_{AB}\vec{S}_A\vec{S}_B$$

magnetic exchange coupling may be decomposed into three main physical contributions: i) the ferromagnetic direct exchange J_0 between the magnetic centres, ii) the antiferromagnetic kinetic exchange contribution ΔJ_{KE} reflecting

the relaxation of the magnetic centres in the low-spin state, and iii) the spin polarisation effects ΔJ_{SP} which corresponds to the differential response of the core (non-magnetic) orbitals in the HS and BS determinants and which may be ferro- or antiferromagnetic, depending on the system.

To extract these contributions from DFT or HF calculations, a set of starting orbital is obtained from an HS state calculation, either in the restricted open-shell formalism (HS.RO) or the quasi-restricted open-shell formalism (HS.qRO). This defines a set of doubly occupied molecular orbitals and a set of singly-occupied orbitals (SOMOs) according to the number of unpaired electrons. The SOMOs are then localised using by default the Pipek-Mezey method resulting in a set of localised magnetic orbitals. By flipping the spin of the electrons associated with one centre, a first BS determinant is obtained for which the energy is computed without optimising any orbital (BS.NOpt). Using the HS.RO and BS.NOpt determinants, the first direct exchange contribution may be calculated as,

$$J_0 = -\frac{E_{\text{HS.RO}} - E_{\text{BS.NOpt}}}{\langle S^2 \rangle_{\text{HS.RO}} - \langle S^2 \rangle_{\text{BS.NOpt}}}$$

Or with the HS.qRO,

$$J_0 = -\frac{E_{\text{HS.qRO}} - E_{\text{BS.NOpt}}}{\langle S^2 \rangle_{\text{HS.qRO}} - \langle S^2 \rangle_{\text{BS.NOpt}}}$$

To compute the second kinetic exchange contribution, from the BS.NOpt determinant, magnetic orbitals are relaxed in the field of the frozen core (non-magnetic) orbitals, resulting in a new BS frozen core determinant (BS.FC). Using the BS.FC and HS.RO or HS.qRO determinants and retrieving the direct exchange contribution, the kinetic exchange may be computed,

$$\Delta J_{\text{KE}} = -\frac{E_{\text{HS.(q)RO}} - E_{\text{BS.FC}}}{\langle S^2 \rangle_{\text{HS.(q)RO}} - \langle S^2 \rangle_{\text{BS.FC}}} - J_0$$

The contribution from the spin polarisation effects is computed by allowing the relaxation of the core orbitals while keeping the magnetic orbitals frozen (FM) from the HS.RO/HS.qRO and the BS.NOpt determinants, resulting in the HS.FM and BS.FM determinants, respectively. Using these two new determinants and retrieving the direct exchange contribution, the spin polarisation effects may be evaluated as,

$$\Delta J_{\text{SP}} = -\frac{E_{\text{HS.FM}} - E_{\text{BS.FM}}}{\langle S^2 \rangle_{\text{HS.(q)RO}} - \langle S^2 \rangle_{\text{BS.Nopt}}} - J_0$$

In addition to the usual use of the Yamaguchi formula with fully relaxed HS and BS determinants, the magnetic exchange coupling may be evaluated by summing these three contributions up, the so-called Recomposition method,

$$J_{\Sigma} = J_0 + \Delta J_{\text{KE}} + \Delta J_{\text{SP}}$$

In order to use the decomposition:

```
%scf DecompositionPath NA,NB
     end
```

with `NA` and `NB` the number of unpaired eletrons in the sites A and B, respectively. The program will then go through several steps to perform the decomposition as described above. To use the HS.RO as reference definition of the orbitals, HFTyp=ROHF must be used whilst, for HS.qRO, it must be HFTyp=UHF.

Caution

Make sure that in your input coordinates "site A" is the site that contains the larger number of unpaired electrons!

5.31 Magnetic Properties Through Quasi Degenerate Perturbation Theory

5.31.1 Quasi Degenerate Perturbation Theory (QDPT) in a nutshell

Quasi Degenerate Perturbation Theory offers a versatile and accurate approach to to a number of magnetic properties for basically every wavefunction-based excited state method.

In a nutshell, at the non-relativistic limit for every excited state single- or multi-reference wavefunction-based method, bearing a CASSCF, MRCI, or a ROCIS type of zeroth order wavefunction, one can set up an excitation problem that is a combination of the zeroth order wavefunction and excited spin-adapted configuration state functions (CSFs) $\left|\Phi_\mu^{SS}\right\rangle$.

That takes the form:

$$\left|\Psi_I^{SS}\right\rangle = \sum\nolimits_\mu C_{\mu l} \left|\Phi_\mu^{SS}\right\rangle \tag{5.173}$$

Here, the upper indices SS stand for a wavefunction of the spin quantum number S and spin projection $M_S = S$. Since the BO Hamiltonian does not contain any complex-valued operator, the solutions $\left|\Psi_I^{SS}\right\rangle$ may be chosen to be real-valued.

Obtaining a solution to the above eigenvalue problem provides the coefficients with which the CSFs enter into the chosen wavefunction, as well as the eigenstates of the spin-free operator. These eigenstates may be used to expand towards the respective relativistic eigenstates by setting up the relevant quasi-degenerate eigenvalue problem. In fact, the spin-orbit coupling (SOC) and spin-spin coupling (SSC) effects along with the Zeeman interaction can be included by means of the quasi-degenerate perturbation theory (QDPT). In this approach, the SOC, SSC, and Zeeman operators are calculated in the basis of pre-selected solutions of the BO Hamiltonian $\left\{\Psi_I^{SM}\right\}$.

$$\left\langle \Psi_I^{SM} \middle| \hat{H}_{\text{BO}} + \hat{H}_{\text{SOC}} + \hat{H}_{\text{SSC}} + \hat{H}_{\text{Z}} \middle| \Psi_J^{S'M'} \right\rangle = \delta_{IJ}\delta_{SS'}\delta_{MM'}E_I^{(S)} + \left\langle \Psi_I^{SM} \middle| \hat{H}_{\text{SOC}} + \hat{H}_{\text{SSC}} + \hat{H}_{\text{Z}} \middle| \Psi_J^{S'M'} \right\rangle \tag{5.174}$$

Diagonalization of this matrix yields the energy levels and eigenvectors of the coupled states. These eigenvectors in fact represent linear combinations of the solutions of $\hat{H}_{\text{BO}}$ with complex coefficients.

The effective one-electron SOC operator in second quantized form can be written as [715]:

$$\hat{H}_{\text{SOMF}} = \frac{1}{2}\sum_{pq} z_{pq}^{-}\hat{a}_p^{\uparrow}\hat{b}_q + z_{pq}^{+}\hat{b}_p^{\uparrow}\hat{a}_q + z_{pq}^{0}\left[\hat{a}_p^{\uparrow}\hat{a}_q - \hat{b}_p^{\uparrow}\hat{b}_q\right] \tag{5.175}$$

Here, $\hat{a}_p^{\uparrow}$ and $\hat{b}_p^{\uparrow}$ stand for creation of α and β electrons, respectively; $\hat{a}_p$ and $\hat{b}_p$ represent the corresponding annihilation operators. The matrix elements $z_{pq}^{-} = z_{pq}^{x} - iz_{pq}^{y}$, $z_{pq}^{+} = z_{pq}^{x} + iz_{pq}^{y}$, and $z_{pq}^{0} = z_{pq}^{z}$ (upper x, y, z indices denote the Cartesian components) are constructed from the matrix elements described in section *Zero-Field Splitting*.

In this concept, the SOC Hamiltonian reads:

$$\left\langle \Phi_I^{SM} \middle| \hat{H}_{\text{SOC}} \middle| \Phi_J^{S'M'} \right\rangle = \sum_{m=0,\pm1} (-1) \left(\begin{array}{cc|c} S' & 1 & S \\ M' & m & M \end{array}\right) \underbrace{\left\langle \Phi_I^{SS} \middle| \middle| H_{-m}^{SOC} \middle| \middle| \Phi_J^{SS} \right\rangle}_{Y_{II'}^{SS'}(m)} \tag{5.176}$$

where m represents the standard vector operator components. $\left(\begin{array}{cc|c} S' & 1 & S \\ M' & m & M \end{array}\right)$ is a Clebsch–Gordon coefficient that has a single numerical value that is tabulated. It satisfies certain selection rules and contains all of the M-dependence of the SOC matrix elements. The quantity $Y_{II'}^{SS'}(m)$ is a reduced matrix element. It only depends on the standard components of the two states involved. There are only three cases of non-zero $Y_{II'}^{SS'}(m)$, which arise from state pairs that either have the same total spin or differ by one unit.[656]

The SSC Hamiltonian reads:

$$\hat{H}_{\text{SSC}} = -\frac{3g_e^2\alpha^2}{8}\sum_{i\neq j}\sum_{m=0,\pm1,\pm2}\frac{(-1)^m}{r_{ij}^5}\left[\mathbf{r}_{ij}\times\mathbf{r}_{ij}\right]_{-m}^{(2)}\left[\mathbf{S}(i)\times\mathbf{S}(j)\right]_m^{(2)} \tag{5.177}$$

For matrix elements between states of the same multiplicity, it can be simplified to

$$\left\langle aSM \left| \hat{H}_{\text{SSC}} \right| a'SM' \right\rangle = \frac{\sqrt{(S+1)(2S+3)}}{\sqrt{S(2S-1)}} \times \sum_m (-1)^m \left(\begin{matrix} S' & 2 \\ M' & m \end{matrix} \middle| \begin{matrix} S \\ M \end{matrix} \right) \sum_{pqrs} D^{(-m)}_{pqrs} \left\langle aSS \left| Q^0_{pqrs} \right| a'SS \right\rangle \tag{5.178}$$

Here,

$$Q^{(0)}_{pqrs} = \frac{1}{4\sqrt{6}} \left\{ E_{pq}\delta_{sr} - S^z_{ps}S^z_{rq} + \frac{1}{2}\left(S^z_{pq}S^z_{rs} - E_{pq}E_{rs} \right) \right\} \tag{5.179}$$

represents the two-electron quintet density. The operators $E_{pq} = \hat{a}^\uparrow_p \hat{a}_q + \hat{b}^\uparrow_p \hat{b}_q$ and $S^z_{pq} = \hat{a}^\uparrow_p \hat{a}_q - \hat{b}^\uparrow_p \hat{b}_q$ symbolize the one-electron density operator and the spin density operator, respectively. The spatial part

$$D^{(0)}_{pqrs} = \frac{1}{\sqrt{6}} \iint \varphi_p(\mathbf{r}_1)\varphi_r(\mathbf{r}_2) \frac{3r_{1z}r_{2z} - \mathbf{r}_1\mathbf{r}_2}{r^5_{12}} \varphi_q(\mathbf{r}_1)\varphi_s(\mathbf{r}_2)\, d\mathbf{r}_1 d\mathbf{r}_2 \tag{5.180}$$

denotes the two-electron field gradient integrals. These two-electron integrals can be evaluated using the RI approximation.

Finally, the Zeeman Hamiltonian is included in the form of:

$$\hat{H}_Z = \mu_B \left(\hat{\mathbf{L}} + g_e \hat{\mathbf{S}} \right) \mathbf{B} \tag{5.181}$$

with $\hat{\mathbf{L}}$ representing the total orbital momentum operator, and $\hat{\mathbf{S}}$ being the total spin operator.

In this concept, the solution of a selected relativistic Hamiltonian provides access to numerous magnetic properties, namely EPR properties (*EPRNMR - keywords for magnetic properties*) and magnetization and susceptibility properties (*Magnetization and Magnetic Susceptibility*). Additionally, monitoring the impact of an external magnetic field to the relativistic eigenstates and eigenvectors (*Addition of Magnetic Fields*) becomes straightforward.

Collectively within the QDPT framework, the following magnetic properties become available:

- Common QDPT magnetic properties

  (1) g-Tensor/Matrix
  (2) D-Tensor/Matrix (Zero Field Splitting, ZFS)
  (3) A-Tensor/Matrix (Hyperfine, HFC)
  (4) Electric Field Gradient
  (5) Magnetization
  (6) Susceptibility
  (7) Inclusion of Magnetic Fields

- Special treatments also include

  (8) g-Tensor/Matrix and A-Tensor/Matrix using the 2nd-order sum-over-states approximation
  (9) *Magnetic Relaxation* functionality

5.31.2 Magnetic Properties Through the Effective Hamiltonian

Since both the energies and the wavefunction of the low-lying spin-orbit states are available, the effective Hamiltonian theory can be used to extract EPR parameters such as the full g-, Zero Field Splitting (ZFS) and hyperfine A-tensors.

Provided that the ground state is non-degenerate. By applying this Hamiltonian on the basis of the model space, i.e. the $|S, M_S\rangle$ components of the ground state, the interaction matrix is constructed.

The construction of effective Hamiltonian relies on the information contained in both the energies and the wavefunctions of the low-lying spin-orbit states. Following des Cloizeaux formalism, the effective Hamiltonian reproduces the energy levels of the "exact" Hamiltonian E_k and the wavefunctions of the low-lying states projected onto the model space $\tilde{\Psi}$:

$$\hat{H}_{\text{eff}} |\tilde{\Psi}_k\rangle = E_k |\tilde{\Psi}_k\rangle$$

These projected vectors are then symmetrically orthonormalized resulting in an Hermitian effective Hamiltonian, which can be written as:

$$\hat{H}_{\text{eff}}|\tilde{\Psi}\rangle = \sum_k |S^{-\frac{1}{2}}\tilde{\Psi}_k\rangle E_k \langle S^{-\frac{1}{2}}\tilde{\Psi}_k|$$

The effective interaction matrix obtained by expanding this Hamiltonian into the basis of determinants belonging to the model space, is then compared to the matrix resulted from expanding the model Hamiltonian. Based on a singular value decomposition procedure, all 9 elements of the g-, A- and/or ZFS-tensors may be extracted.

5.31.3 QDPT Keywords

Starting from ORCA 6.0, the calculation of the magnetic properties through the Quasi Degenerate Perturbation Theory (QDPT) in all available correlation-type modules is unified and simplified. Following the general architecture design of ORCA 6.0, the computation of all the involved magnetic properties are centrally performed by a driver data structure called the QDPT Driver. The Driver takes into account all of the specific variables that are populated by the involved module and proceeds accordingly to calculate and represent the requested property in a uniform fashion. This presently involves the `casscf`, `mrci`, `rocis`, and `lft` modules

In this way,

1) the analysis of results from the user's perspective is simplified

2) cross-module correlation and comparisons are easily accessible

The general keywords that activate the generation of QDPT properties are:

```
%METHOD (casscf, mrci, rasci, rocis, lft, ...)
  REL            # the name of the relativistic block
    DoSOC true   # include the SOC contribution
    DoSSC true   # include the SSC contribution
  END
END
```

Starting from ORCA 6.1, the relativistic bock has a common structure across the methods that are connected to the QDPT driver. Hence, `rel` is the default block name for all connected methods. In the case of the `%mrci` block, `soc` is an alternative name to maintain for this release consistency with the old input relativistic blocks. The `soc` block name will be deprecated in the next release, so one is recommended to use the `rel` block name for requesting QDPT properties.

The QDPT keywords are:

```
%METHOD (CASSCF, RASCI, MRCI, ROCIS, LFT)
  REL
   #-------------------------------------------------------
   # SOC/SSC
   #-------------------------------------------------------
   DoSOC true
   DoSSC true
   PrintLevel 4

   #=======================================================
   # MAGNETIC PROPERIES
   #=======================================================

   #-------------------------------------------------------
   #Do Magnetization Magntization/Susceptibily Properties
   #-------------------------------------------------------
   DoMagnetization true
   DoSusceptibility true
   #-------------------------------------------------------
   LebedevPrec 5           # Precision of the grid for different field
```

(continues on next page)

(continued from previous page)

```
nPointsFStep 5          # number of steps for numerical differentiation
MAGFieldStep 100.0      # Size of field step for numerical differentiation
MAGTemperatureMIN 4.0   # minimum temperature (K) for magnetization
MAGTemperatureMAX 4.0   # maximum temperature (K) for magnetization
MAGTemperatureNPoints 1 # number of temperature points for magnetization
MAGFieldMIN 0.0         # minimum field (Gauss) for magnetization
MAGFieldMAX 70000.0     # maximum field (Gauss) for magnetization
MAGNpoints 15           # number of field points for magnetization
SUSTempMIN 1.0          # minimum temperature (K) for susceptibility
SUSTempMAX 300.0        # maximum temperature (K) for susceptibility
SUSNPoints 300          # number of temperature points for susceptibility
SUSStatFieldMIN 0.0     # minimum static field (Gauss) for susceptibility
SUSStatFieldMAX 0.0     # maximum static field (Gauss) for susceptibility
SUSStatFieldNPoints 1   # number of static fields for susceptibility

#-------------------------------------------------------
# Magnetic Field Magnetic Filed Perturbation (Zeeman Effect)
#-------------------------------------------------------
DoMagneticField true
Temperature 10, 50, 300
B 10000, 20000, 30000

#-------------------------------------------------------
# EPR g-Tensor/Matrix
#-------------------------------------------------------
DoGTensor true
NDoubGtensor 2

#-------------------------------------------------------
# EPR D-Tensor/Matrix (Zero Field Splitting, ZFS)
#-------------------------------------------------------
DoDTensor true

#-------------------------------------------------------
# EPR A-Tensor/Matrix (Hyperfine, HFC)
# => Presently only in CASSCF and LFT
#-------------------------------------------------------
DoAMatrix true
AMatrixNuc 0,1

#-------------------------------------------------------
# 2nd-order SUM OVER STATES (g-, A-Tensors/Matrices)
# => Presently only in CASSCF and MRCI
#-------------------------------------------------------
DoSOS True

#-------------------------------------------------------
# ZFS from an excited state multiplet
#-------------------------------------------------------
IStates 4,5,6

#-------------------------------------------------------
# Magnetic Relaxation
# => Presently only in CASSCF and MRCI
#-------------------------------------------------------
DoMagrelax True
projectHSOC True       # Project QDPT ground multiplet (always true)
projectedstates Number # Multiplicity of the ground multipler

#=======================================================
# OPTICAL/X-RAY SPECTROSCOPIES
```

(continues on next page)

(continued from previous page)

```
    #=======================================================

    #-------------------------------------------------------
    # (X)MCD (X-ray) Magnetic Circular Dichroism Spectroscopy
    #-------------------------------------------------------
    DoMCD True

    #-------------------------------------------------------
    # XESSOC X-ray Emission Spectroscopy
    #-------------------------------------------------------
    DoXESSOC True

    #-------------------------------------------------------
    # RIXSSOC X-ray Resonance Inelastic Scattering Spectroscopy
    #-------------------------------------------------------
    DoRIXSSOC True
    DoElastic True
   END
END
```

5.31.4 Organization of QDPT Magnetic Properties Computation

In a first step, SOC contributions will be computed for any level of theory that is available

```
----------------------------------
QDPT WITH CASSCF/NEVPT2/MRCI/LFT... DIAGONAL ENERGIES
----------------------------------


************************************
COMPUTING QDPT HAMILTONIAN
************************************
```

Initially, the SOC part to the splitting is calculated. Firstly, the diagonal (with respect to the spin) matrix element type of

$\langle\Psi_I|\sum_{pq}\mathbf{z}^x_{pq}S^z_{pq}|\Psi_J\rangle$, $\langle\Psi_I|\sum_{pq}\mathbf{z}^y_{pq}S^z_{pq}|\Psi_J\rangle$, $\langle\Psi_I|\sum_{pq}\mathbf{z}^z_{pq}S^z_{pq}|\Psi_J\rangle$

are evaluated between states of the same multiplicity and

$\langle\Psi_I|\sum_{pq}\mathbf{z}^x_{pq}S^+_{pq}|\Psi_J\rangle$, $\langle\Psi_I|\sum_{pq}\mathbf{z}^y_{pq}S^+_{pq}|\Psi_J\rangle$, $\langle\Psi_I|\sum_{pq}\mathbf{z}^z_{pq}S^+_{pq}|\Psi_J\rangle$

between states of different multiplicities.

Requesting `PrintLevel > 3` prints this table:

```
---------------------------------------------------------------------------
                        CALCULATED REDUCED SOC MATRIX ELEMENTS
---------------------------------------------------------------------------
        Block                Root         <I|Lx S |J> <I|Ly S |J> <I|Lz S |J>
 I(Mult)     J(Mult)      I         J         cm-1        cm-1        cm-1
---------------------------------------------------------------------------
 0( 3)       0( 3)        0         0         0.00       -0.00       -0.00
 0( 3)       0( 3)        1         0        -0.00       -0.00        0.00
 0( 3)       0( 3)        1         1         0.00       -0.00       -0.00
 0( 3)       0( 3)        2         0         0.00       -0.00       -0.00
 0( 3)       0( 3)        2         1      -199.78      174.81     -142.34
...

 0( 3)       1( 1)        0         1        -0.67        0.59       -0.48
```

(continues on next page)

(continued from previous page)

```
 0( 3)        1( 1)         0         2       -157.31       137.65      -112.09
 0( 3)        1( 1)         0         3         -0.00        -0.00        -0.00
 0( 3)        1( 1)         0         4        -79.51       -33.73        70.17
 0( 3)        1( 1)         0         5         24.77        84.05        68.46
...
```

```
*************************************
Doing QDPT with ONLY SOC!
*************************************
```

Next, the non-zero SOC Matrix Elements will be printed

```
-----------------------------------
NONZERO SOC MATRIX ELEMENTS (cm**-1)
-----------------------------------

           Bra                        Ket
<Block Root   S    Ms   | HSOC |  Block Root   S    Ms>    =  Real-part     Imaginary␣
↪part
-------------------------------------------------------------------------------------
↪---
   0    2  1.0  1.0              0     1  1.0  1.0            0.000            -71.
↪172
   0    3  1.0  1.0              0     2  1.0  1.0            0.000              1.
↪542
   0    4  1.0  1.0              0     0  1.0  1.0            0.000             50.
↪048
   0    5  1.0  1.0              0     0  1.0  1.0            0.000            -48.
↪827
   0    5  1.0  1.0              0     4  1.0  1.0            0.000            -40.
↪119
   0    6  1.0  1.0              0     1  1.0  1.0            0.000             -0.
↪197
   0    6  1.0  1.0              0     3  1.0  1.0            0.000              8.
↪724
   0    7  1.0  1.0              0     1  1.0  1.0            0.000             -2.
↪695

...

   1    0  0.0  0.0              0     4  1.0  1.0           16.671            -39.
↪318
   1    0  0.0  0.0              0     5  1.0  1.0          -41.569             12.
↪256
   1    0  0.0  0.0              0     4  1.0  0.0           -0.000            -49.
↪087
   1    0  0.0  0.0              0     5  1.0  0.0           -0.000            -47.
↪869
   1    0  0.0  0.0              0     4  1.0 -1.0           16.671             39.
↪318
   1    0  0.0  0.0              0     5  1.0 -1.0          -41.569            -12.
↪256
   1    1  0.0  0.0              0     0  1.0  1.0            0.294              0.
↪336
   1    1  0.0  0.0              0     4  1.0  1.0           41.742            -12.
↪307
/
...
```

followed by the printing of the SOC Hamiltonian

```
Note: In the following the full <I|HBO+SOC|J> are printed in the CI Basis.
      I,J are compound indices for |Block/Mult, Ms, Root>, where the states
      are ordered first by MultBlock, then Ms and finally Root.
-----------------
SOC MATRIX (A.U.)
-----------------
```

Finally, the relativistically corrected eigenvalues and eigenvectors are printed

```
Lowest eigenvalue of the SOC matrix:    -149.86223277 Eh
Energy stabilization:    -2.54512 cm-1
Eigenvalues:     cm-1         eV      Boltzmann populations at T =  300.000 K
   0:            0.00        0.0000       3.36e-01
   1:            2.37        0.0003       3.32e-01
   2:            2.37        0.0003       3.32e-01
   3:         7757.65        0.9618       2.33e-17
   4:         7757.66        0.9618       2.33e-17
   5:        11913.81        1.4771       5.15e-26

...

The threshold for printing is 0.0100
Eigenvectors:
                         Weight      Real          Image    : Block Root    Spin
↪Ms
 STATE   0:       0.0000
                         0.388265     0.410320    -0.468937 :     0    0    1    1
                         0.223270    -0.000000     0.472514 :     0    0    1    0
                         0.388265     0.410320     0.468937 :     0    0    1   -1
 STATE   1:       2.3703
                         0.310686     0.534586     0.157809 :     0    0    1    1
                         0.378606     0.000017    -0.615309 :     0    0    1    0
                         0.310706     0.534623    -0.157747 :     0    0    1   -1
 STATE   2:       2.3703
                         0.300970    -0.214003    -0.505146 :     0    0    1    1
                         0.398078    -0.000007    -0.630934 :     0    0    1    0
                         0.300949    -0.214019     0.505119 :     0    0    1   -1

...
```

Now, all relevant QDPT properties are calculated and printed:

```
************************************
COMPUTING QDPT PROPERTIES
************************************
```

(1) For g-Tensor/Matrix

```
---------------------------------------------
ELECTRONIC G-MATRIX FROM EFFECTIVE HAMILTONIAN
---------------------------------------------
```

(2) For ZFS (D-Tensor/Matrix) on the basis of the 2nd-order and Effective Hamiltonian approximations

```
-------------------------------------------
           ZERO-FIELD SPLITTING
2ND ORDER SOC CONTRIBUTION
-------------------------------------------
-------------------------------------------------------
                  ZERO-FIELD SPLITTING
EFFECTIVE HAMILTONIAN SOC CONTRIBUTION
-------------------------------------------------------
```

(3) For HFC (A-Tensor/Matrix)

```
-------------------------
QDPT HFC A-MATRICES
-------------------------
```

In the CASSCF and LFT modules, one may request the A-Matrix using, for example:

```
AMatrix true
AMatrixNuc 0, 2
```

If requested, one can specify isotopes for the calculations

```
AMatrix true
AMatrixNuc 0, 2, 3

#e.g. for Nuclei 0, this requires partially or fully populating the following␣
↪quantities:

AMATRIXP[0]=125.6   #the P-value for the hyperfine ge*gN*beta*betaN
AMATRIXQ[0]=3.87    #the quadrupole value (in barn)
AMATRIXIP[0]=2.5    #spin of the isotope for the A-tensor
AMATRIXIQ[0]=0.0    #spin of the isotope of the Q-tensor
AMATRIXTP[0]=237    #AMatrixISTPP[0]
AMATRIXTQ[0]=0      #AMatrixISTPQ[0]

# It follows that for Nuclei 2, one does
AMATRIXTP[1]=36
...
# and so on.
```

(4) For Electric Field Gradient Tensor

```
---------------------------------------
EFG TENSOR
---------------------------------------
```

(5) & (6) For Magnetization/Susceptibility

```
--------------------------------------------------
SOC CORRECTED MAGNETIZATION AND/OR SUSCEPTIBILITY
--------------------------------------------------
```

(7) For External Magnetic Fields Contributions

```
-------------------------------------------------------------------------------
↪------------------------
               SOC TRANSITION MAGNETIC DIPOLE CONTRIBUTIONS IN EXTERNAL MAGNETIC␣
↪FIELD
           Magnetic field Bx =       1.00 Gauss  By =       0.00 Gauss  Bz =      ␣
↪0.00 Gauss
-------------------------------------------------------------------------------
↪------------------------
 States       Energy      Energy     Osh.Str           M2            MX           ␣
↪ MY             MZ
              (cm-1)        (eV)        (au)          (au**2)        (au)          ␣
↪(au)            (au)
-------------------------------------------------------------------------------
↪------------------------
 0  0         0.00       0.0000     0.00000000     0.00000145     0.00090159     0.
↪00061858     0.00050413
 0  1        20.20       0.0025     0.00000000     0.00000126     0.00042808     0.
```

(continues on next page)

(continued from previous page)

```
↪00067630     0.00078545
 1  1         0.00       0.0000     0.00000000     0.00000000     0.00003430     0.
↪00002347     0.00001909
-------------------------------------------------------------------------------
↪-----------------------
```

Following this, all relativistically corrected optical spectra are printed under the same correction scheme (as Discussed in *One Photon Spectroscopy*).

```
-------------------------------------------------------------------------------
↪---------------------
      SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪---------------------
      Transition         Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                          (eV)      (cm-1)    (nm)   (*population)  (au**2)    (au)␣
↪     (au)      (au)
-------------------------------------------------------------------------------
↪---------------------

...
```

(8) Sum-over-states g- and A-tensors/matrices (for now, available only in CASSCF and MRCI) get printed in much the same way as those of the Effective Hamiltonian. The main difference here is the designator:

```
-------------------------------------------------------------------------
SUM OVER STATES CALCULATION OF THE SPIN HAMILTONIAN (for g and HFC tensors)
-------------------------------------------------------------------------
```

The matrix elements that are used in the summation are printed after

```
-----------------------
MATRIX ELEMENT PRINTING
-----------------------
```

The g-tensor additionally prints out the contribution of each state to the `g(OZ/SOC)` term.

```
------------------------------
BREAKDOWN OF g(OZ/SOC) BY STATE
------------------------------

STATE                                g1           g2           g3
-----------------------------------------------------------------
```

Note that for CASSCF, *CASSCF Linear Response* without the orbital response is equivalent to the sum-over-states when including all states. As such, the main draw of the sum-over-states is this state-by-state breakdown.

If the SSC is requested, Spin-Spin Coupling contributions will be generated in a second step and will be added to the SOC Hamiltonian to generate SOC+SSC contributions

```
-----------------------------------------
Calculating Spin-Spin Coupling Integrals
-----------------------------------------
```

The program will then undergo the exact same analysis as above, printing the SOC+SSC analysis.

```
************************************************************
* DOING EVERYTHING A SECOND TIME: THIS TIME INCLUDING SSC *
************************************************************
```

(continues on next page)

(continued from previous page)

```
************************************
COMPUTING QDPT HAMILTONIAN
************************************

************************************
Doing QDPT with SOC AND SSC!
************************************

-------------------------------------
NONZERO SOC and SSC MATRIX ELEMENTS (cm**-1)
-------------------------------------

          Bra                        Ket
<Block Root  S    Ms  | HSOC + HSSC |  Block Root  S    Ms>    =  Real-part    ␣
↪Imaginary part
-------------------------------------------------------------------------------
↪---
  0    2  1.0  1.0                      0     1  1.0  1.0          -0.000       ␣
↪   -71.172
  0    3  1.0  1.0                      0     1  1.0  1.0          -0.001       ␣
↪     0.000
  0    3  1.0  1.0                      0     2  1.0  1.0           0.020       ␣
↪     1.542
  0    4  1.0  1.0                      0     0  1.0  1.0          -0.222       ␣
↪    50.048
  0    5  1.0  1.0                      0     0  1.0  1.0          -0.228       ␣
↪   -48.827
  0    5  1.0  1.0                      0     4  1.0  1.0           0.332       ␣
↪   -40.119
  0    6  1.0  1.0                      0     1  1.0  1.0          -0.030       ␣
↪    -0.197
```

This is done for both the magnetic properties, e.g. the ZFS,

```
-------------------------------------------------------
                  ZERO-FIELD SPLITTING
EFFECTIVE HAMILTONIAN SOC and SSC CONTRIBUTION
-------------------------------------------------------
```

as well as the optical properties

```
-------------------------------------------------------------------------------
↪---------------------
  SOC+SSC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪---------------------
     Transition         Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                         (eV)      (cm-1)    (nm)   (*population)  (au**2)    (au)␣
↪     (au)      (au)
-------------------------------------------------------------------------------
↪---------------------
```

5.32 Magnetic Relaxation

5.32.1 General description of the program.

For magnetic molecules, ORCA now offers the possibility of computing the magnetic relaxation times. This is done using the Orca_Magrelax module. In general, for a system with electronic and vibrational degrees of freedom, the Hamiltonian can be written as [740, 741]:

$$\begin{aligned}
\hat{H} &= \hat{H}_{spin} + \hat{H}_{ph} + \hat{H}_{sph} \\
\hat{H}_{sph} &= \sum_{\alpha} \frac{\partial \hat{H}_{spin}}{\partial Q_\alpha} \hat{Q}_\alpha + \frac{1}{2} \sum_{\alpha\beta} \left(\frac{\partial^2 \hat{H}_{spin}}{\partial Q_\alpha \partial Q_\beta} \right) \hat{Q}_\alpha \hat{Q}_\beta + \dots \\
&= \sum_{\alpha} \hat{V}_\alpha \hat{Q}_\alpha + \frac{1}{2} \sum_{\alpha\beta} \hat{V}_{\alpha,\beta} \hat{Q}_\alpha \hat{Q}_\beta + \dots \\
\hat{H}_{ph} &= \sum_{\alpha} \hbar\omega_\alpha \left(\hat{n}_\alpha + \frac{1}{2} \right)
\end{aligned}$$

Here $\hat{H}_{spin}$ is the electronic Hamiltonian and $\hat{H}_{ph}$ is the phononic (vibrational) Hamiltonian. These first two terms constitute the Born-Oppenheimer approach and if this all that the hamiltonian contained then the total wavefunction would just be the product of the electronic and phononic wavefunctions. The $\hat{H}_{sph}$ term is the term that introduces interaction between the electronic and phononic parts and makes the total wavefunction a linear combination of the aforementioned product states.

The phononic Hamiltonian can be approximated as a harmonic oscillator. The coupling $\hat{H}_{sph}$ is usually expressed as a Herzberg-Teller expansion of $\hat{H}_{spin}$. The $\hat{H}_{spin}$ has been expressed in literature [742, 743] as a parametric spin hamiltonian and more recently as a CASSCF Hamiltonian [744]. In our present methodology, the QDPT Hamiltonian is used as $\hat{H}_{spin}$(*Magnetic Properties Through Quasi Degenerate Perturbation Theory*). This allows for a parameter-free first-principles estimation of magnetic relaxation. The derivatives of the QDPT matrix are numerically computed. We use the Fermi golden rule to compute the elements of the rate matrix.

$$P_{FI} = \frac{2\pi}{\hbar} \int |\langle F|\hat{H}^{\mu}_{pert}|I\rangle|^2 \delta(E_F - E_I)\rho(E_F) dE_F$$

Where the $\hat{H}^{\mu}_{pert}$ is the μ-order perturbation theory expansion of the transition operator. It is described by the expression[745]:

$$\hat{H}^{\mu}_{pert} = \sum_{\mu=1}^{\infty} \hat{H}_{sph} (\frac{1}{E_I - H_0 + i(0)^+} \hat{H}_{sph})^{(\mu-1)}$$

Thus we see that our implmentation depends on the two infinite order expansions that must be truncted in a finite manner for practical considerations. The first is the Herzberg-Teller expansion and the second is the perturbation theory expression.

Fig. 5.65: Mechanism of energy exchange between the electronic and phononic degrees of freedom.

If we truncate both of these at the first order, we recover expressions for the one-phonon process, also known as the Orbach Relaxation. If one of these expressions is truncated to the second order, we get a component of the two-phonon processes collectively known as the Raman processes. The second-order Herzberg-Teller gives the so-called Raman I component, and the second-order perturbation theory gives the Raman II component.

The Raman I process is rapid and oscillatory and is presently assumed to average out to zero in our implementation [746]. This leaves the Orbach process, which dominates magnetic relaxation at higher temperatures, and the Raman Process, which dominates magnetic relaxation at lower temperatures. The resulting rate equations for Orbach process are as follows [747]:

$$\Gamma_{FI}^{-} = \frac{\pi}{2\hbar} \sum_{\alpha} \int |V_{FI}^{\alpha}|^2 \bar{n}(\hbar\omega) \delta(\Delta E_{FI} - \hbar\omega) \rho_{\alpha}(\hbar\omega) d\hbar\omega$$

$$\Gamma_{Fi}^{+} = \frac{\pi}{2\hbar} \sum_{\alpha} \int |V_{FI}^{\alpha}|^2 (\bar{n}(\hbar\omega) + 1) \delta(\Delta E_{FI} + \hbar\omega) \rho_{\alpha}(\hbar\omega) d\hbar\omega$$

and the Raman rate equations become[748]:

$$\Gamma_{FI}^{--} = \frac{\pi}{\hbar}\sum_{\alpha,\beta}\int\int|\sum_{C}\frac{V_{FC}^{\alpha}}{\Delta E_{CI}-\hbar\omega'}+\frac{V_{CI}^{\alpha}}{\Delta E_{CI}-\hbar\omega}|^2$$
$$\bar{n}(\hbar\omega)\bar{n}(\hbar\omega')\delta(\Delta E_{FI}-\hbar\omega-\hbar\omega')\rho_{\alpha}(\hbar\omega)\rho_{\beta}(\hbar\omega')d\hbar\omega d\hbar\omega'$$
$$\Gamma_{FI}^{++} = \frac{\pi}{\hbar}\sum_{\alpha,\beta}\int\int|\sum_{C}\frac{V_{FC}^{\alpha}}{\Delta E_{CI}+\hbar\omega'}+\frac{V_{CI}^{\alpha}}{\Delta E_{CI}+\hbar\omega}|^2$$
$$(\bar{n}(\hbar\omega)+1)(\bar{n}(\hbar\omega')+1)\delta(\Delta E_{FI}+\hbar\omega+\hbar\omega')\rho_{\alpha}(\hbar\omega)\rho_{\beta}(\hbar\omega')d\hbar\omega d\hbar\omega'$$
$$\Gamma_{FI}^{-+} = \frac{\pi}{\hbar}\sum_{\alpha,\beta}\int\int|\sum_{C}\frac{V_{FC}^{\alpha}}{\Delta E_{CI}-\hbar\omega'}+\frac{V_{CI}^{\alpha}}{\Delta E_{CI}+\hbar\omega}|^2$$
$$(\bar{n}(\hbar\omega)+1)\bar{n}(\hbar\omega')\delta(\Delta E_{FI}-\hbar\omega+\hbar\omega')\rho_{\alpha}(\hbar\omega)\rho_{\beta}(\hbar\omega')d\hbar\omega d\hbar\omega'$$
$$\Gamma_{FI}^{+-} = \frac{\pi}{\hbar}\sum_{\alpha,\beta}\int\int|\sum_{C}\frac{V_{FC}^{\alpha}}{\Delta E_{CI}+\hbar\omega'}+\frac{V_{CI}^{\alpha}}{\Delta E_{CI}-\hbar\omega}|^2$$
$$\bar{n}(\hbar\omega)(\bar{n}(\hbar\omega')+1)\delta(\Delta E_{FI}+\hbar\omega-\hbar\omega')\rho_{\alpha}(\hbar\omega)\rho_{\beta}(\hbar\omega')d\hbar\omega d\hbar\omega'$$

Here, $V_{FI}^{\alpha} = \langle F|\hat{V}^{\alpha}|I\rangle$ is a matrix element of the linear spin-phonon coupling for the mode α on the electronic basis. The convention $\alpha \geq \beta$ for the phonon modes is a convenient choice. This convention introduces a factor of $(1-(\delta_{\alpha\beta}/2))$ inside the first summation. The diagonal element of the rate matrix is determined by $\Gamma_{II} = \sum_{F\neq I} -\Gamma_{FI}$. This rate matrix can be diagonalised to give eigenvalues that are $-\tau_k^{-1}$ for each mode of relaxation k. The lowest eigenvalue corresponds to the system being in equilibrium. The second lowest value is the magnetic relaxation rate.

5.32.2 Computing magnetic relaxation times using Orca_Magrelax.

Orca_Magrelax can be used either from an Orca input or as a standalone module. The following example shows how such a calculation can be run from the ORCA main program for a $[DyF_2]^{2+}$ toy system. 'dyf2casoptfreq' is a separately converged geometry optimisation + hessian calculation in ORCA, and 'dyf2cas' is a converged CASSCF calculation on the optimised geometry. The use of magrelax inside orca requires the following additions to a typical CASSCF calculation. Inside the CASSCF block, 'projectHSOC' must be set to true, and the keyword 'projectedstates' must be set to the multiplicity of the lowest multiplet of the magnetic system. In the example, the lowest multiplet is $^6H_{15/2}$ state, which has a multiplicity of 16. The keyword 'domagrelax' must be set to true in the rel block of CASSCF. The '%qgprop' block tells the program to compute numerical derivatives for the QDPT matrix. Then the magrelax block itself provides additional details on the computation of the magnetic relaxation rates.

```
!DKH2 DKH-def2-TZVP AutoAux moread
%moinp "dyf2cas.gbw"  #--------------------converged CASSCF single point␣
↪calculation

#-----------------------------------------------------------------------------------
↪-----------------------------------------------
#picture change effects for DKH2
#-----------------------------------------------------------------------------------
↪-----------------------------------------------
%rel
 picturechange 2
end

#-----------------------------------------------------------------------------------
↪-----------------------------------------------
#SARC2 basis for the Dy atom
#-----------------------------------------------------------------------------------
↪-----------------------------------------------
%basis
NewGTO Dy "SARC2-DKH-QZV" end
NewAuxGTO Dy  "SARC/J" end
```

(continues on next page)

(continued from previous page)

```
end

#-----------------------------------------------------------------------------------
↪---------------------------------------------
#CASSCF settings
#-----------------------------------------------------------------------------------
↪---------------------------------------------
%casscf
        nel 9 #---------------------------number of electrons in valence
        norb 7 #--------------------------number of orbitals in valence
        mult 6 #--------------------------list of multiplicities to be considered
        nroots 21 #------------------------number of roots corresponding to each␣
↪multiplicity
        trafostep RI #---------------------Use RI
        maxiter 400 #----------------------maximum iterations for CASSCF
        projectHSOC true #-----------------project H_SOC = QDPT matrix to the␣
↪lowest multiplet
        projectedstates 16 #---------------number of states in the lowest multiplet
        rel
        dosoc true #-----------------------enables generation of QDPT matrix
        domagrelax true #------------------stores data from the QDPT driver needed␣
↪for magrelax
        end
end

#-----------------------------------------------------------------------------------
↪---------------------------------------------
#quantized geometry properties module (generates derivatives of H_SOC)
#-----------------------------------------------------------------------------------
↪---------------------------------------------
%qgprop
MODE DERIV #-----------------------------compute derivatives
COORDSTYPE ENWNORMAL #---------------------displace energy weighted distance along␣
↪normal modes
STEPSIZE 0.00005 #-------------------------base displacement
HESSIAN "dyf2casoptfreq.hess" #------------hessian file
DERHSOC true #----------------------------property for which derivatives are␣
↪needed (H_SOC)
end

#-----------------------------------------------------------------------------------
↪---------------------------------------------
#magrelax settings
#-----------------------------------------------------------------------------------
↪---------------------------------------------
%magrelax
magfld 0.0, 0.0,0.01 #---------------------magnetic field in x, y, z directions␣
↪(in Tesla)
#inputfile "dyf2test_magrelax2_maxsoc" #---use if casscf and qgprop blocks are in␣
↪a separate file
HESSIAN "dyf2casoptfreq.hess" #------------hessian file
temperature 2,150,1 #----------------------temperature range start, finish, step␣
↪size (K)
degenThresh 0.1 #--------------------------energy threshold for detecting␣
↪degeneracy (in cm-1)
doRaman false #----------------------------orbach is always true raman can be␣
↪selected
gridMax 8000 #-----------------------------upper limit of integration (Raman only)
gridMin 0 #--------------------------------lower limit of integration (Raman only)
nGrid 2000 #-------------------------------no of integration grid points of␣
↪integration (Raman only)
```

(continues on next page)

(continued from previous page)

```
keepData true #---------------------------keep coupling constants for future runs
end

#-------------------------------------------------------------------------------
↪---------------------------------------------
#optimized geometry
#-------------------------------------------------------------------------------
↪---------------------------------------------
*xyzfile 1 6 dyf2casoptfreq.xyz
```

Every time magrelax is run from Orca, it generates a magrelax input file which can be modified to run for different experimental conditions. It will read the files generated by the previous run. The file contains all the keywords that can be called from within ORCA, described with the comments next to each line. This file can be run from the command line as 'orca_magrelax dyf2.magrelaxinp' as long as the main orca program is in the path directory.

```
none #DMAGRELAXINP InputFile
dyf2casoptfreq.hess #DMAGRELAXINP Hessfile
2.000000 150.000000 1.000000 #DMAGRELAXINP templist start, stop, stepsize
0.00000000000000000000 0.00000000000000000000 0.01000000000000000021 #DMAGRELAXINP␣
↪magfld x, y, z
1 0 1 0#DMAGRELAXINP ratemodes, doRaman, keepData, debug
0.10000000000000000555 #DMAGRELAXINP degenThresh
1000.000000 0.000000 1000#DMAGRELAXINP gridmax, gridmin, ngrid
10.000000 #DMAGRELAXINP gamma
```

5.32.3 Running Orca_Magrelax calculations in various modules

It is also possible to compute magnetic relaxation in other modules, which produce a QDPT matrix in ORCA, such as NEVPT2 or HQD-NEVPT2. These can be simply enabled in the CASSCF and they will be automatically used for the magrelax calculations. For calculations that involve the %mrci block the magrelax keywords must not be put in the CASSCF block and instead included in the respective mrci input block. If calculation such as MREOM, MRCEPA or MRACPF that requires an mrci block is run then the magrelax related keywords would still go into the mrci block.

```
%casscf
 nel 9
 norb 7
 trafostep RI
 maxiter 400
 mult 6
 nroots 21
 rel
  dosoc true
 end
end

%mrci
 newblock 6 *
  nroots 21
  refs cas(9,7) end
  excitations cis
 end
 soc
  dosoc true
  domagrelax true
  projectHSOC true
  projectedstates 16
 end
end
```

5.32.4 Fixing common issues

- Single point crashes: QGPROP will generate 2 geometries per vibrational mode. This can result in $(3N-6)\times 2$ geometries which have to be run in single point. These are named '<Basename>.geom-<geometry index>'. A user can rerun any geometry that didn't run with modified settings if needed.
- Modified settings don't show up in computation of linear spin phonon coupling constants: Each time QGPROP is run, it looks if the displaced geometries have converged from a previous run. If converged files are found, they are not recomputed to save on compute time. If the data of a previous run is not need by user 'keepData false' in the magrelax block will delete magrelax and qgprop related data after each run.
- Integration Grid: The Raman computation involves a numerical integration over phonon energies. The upper limit, lower limit and the number of grid points can be selected using the keywords 'gridMax', 'gridMin' and 'nGrid'.
- Differentiation grid: Since linear derivatives are calculated numerically, the step size given to qgprop can have influence the numerical stability of the results. Currently, the displacement of each vibrational mode is weighted by the reciprocal of energy. The 'Stepsize' keyword in qgprop can be used to scale a base displacement of 1 Angstrom.
- Magnetic Field misaligned: Current implmentation requires the user to input the magnetic field to be parallel with the axis of anisotropy. The ground KD g-tensor in CASSCF provides a rotation matrix that can be used to rotate from the molecular frame z-axis.

5.32.5 Magrelax Keywords

Table 5.20: `%method` block input keywords relevant for Magrelax calculations

Keyword	Options	Description
`domagrelax`	`true`	Store data needed for magrelax
`projectHSOC`	'true'	project QDPT ground multiplet (always true)
`projectedstates`	number	multiplicity of ground multipler

Table 5.21: `%qgprop` block input keywords relevant for Magrelax calculations

Keyword	Options	Description
`MODE`	`DERIV`	Calculate derivatives
`COORDSTYPE`	`ENWNORMAL`	Use energy weighted displacements for normal modes
`STEPSIZE`	number	scaling of the unweighted step size
`HESSIAN`	name	hessian file name
`DERHSOC`	`true`	Calculate derivatives of the QDPT hamiltonian with SOC

Table 5.22: `%magrelax` block input keywords relevant for Magrelax calculations

Keyword	Options	Description
magfld	vector	magnetic field (T)
inputfile	name	input file name (no file extension) for data if casscf and qgprop is skipped (optional)
HESSIAN	name	hessian file name
temperature	start, end, step	temperature start,end, step size (K)
degenThresh	threshold	threshold of degeneracy (states within this are ignored) (cm-1)
doRaman	'true'/'false'	include raman relaxation (orbach always on)
gridMax	number	upper limit for numerical integration (raman only)
gridMin	number	lower limit for numerical integration (raman only)
nGrid	number	number of grid points for numerical integration (raman only)
keepData	'true'/'false'	keepData generated by magrelax for future run

5.33 Simulation of (Magnetic) Circular Dichroism and Absorption Spectra

5.33.1 General description of the program

ORCA can now simulate optical spectra that include spin-orbit coupling contributions at all levels of theory by using a common implementation. [749]

Following the energy-loss approach, the absorption cross section for a transition between states $\tilde{P}$ and $\tilde{Q}$ can be expressed as:

$$\sigma_{\tilde{P}\tilde{Q}} = \frac{4\pi^2}{c(E_{\tilde{Q}} - E_{\tilde{P}})}|T_{\tilde{P}\tilde{Q}}|^2,$$

where c is the speed of light; $E_{\tilde{P}}$ and $E_{\tilde{Q}}$ are the energy of the states $\tilde{P}$ and $\tilde{Q}$, respectively; $T_{\tilde{P}\tilde{Q}}$ is the transition moment between states $\tilde{P}$ and $\tilde{Q}$ and it can be computed with different expressions based on the applied approximation.

Under a dipolar approximation to the light-matter interaction, the transition moment takes the form:

$$T_{\tilde{P}\tilde{Q}} = \sum_{i=1}^{N} \langle \tilde{P} | \, \mathcal{E} \cdot \hat{\mathbf{p}}_i \, | \tilde{Q} \rangle,$$

where the sum runs over all electrons i; $\hat{\mathbf{p}}_i$ is the linear momentum operator; and $\mathcal{E}$ is the polarization vector of the incident light.

In order to take into account all the electric and magnetic mechanisms in the transition, it is necessary to use the full field-matter interaction operator (FFMIO). For transition moment, it leads to the equation (5.182).

$$T_{\tilde{P}\tilde{Q}} = \frac{e}{m_e} \sum_{i=1}^{N} \langle \tilde{P} | \, \mathcal{E} \cdot \left[e^{i\mathbf{k}\cdot\hat{\mathbf{r}}_i} \hat{\mathbf{p}}_i \right] | \tilde{Q} \rangle \tag{5.182}$$

where $\hat{r}_i$ is the position operator of i-th electron and $\mathbf{k}$ is the wave vector that points in the direction of the light propagation whose magnitude is related to the wavelength by $\lambda = 2\pi/|\mathbf{k}|$.

For free-rotating molecules, it is necessary to consider all possible orientations of the molecule with respect to the direction of the incident light. In some cases, such as the absorption of linear-polarized light under a dipolar approximation, the effect of the orientation can be averaged analytically. However, numerical integration over some selected molecular orientations, labeled as o in equation (5.183), is generally necessary.

$$< \sigma_{\tilde{P}\tilde{Q}} > = \sum_{o} w_o \frac{4\pi^2}{c(E_{\tilde{Q}}(o) - E_{\tilde{P}}(o))}|T_{\tilde{P}\tilde{Q}}(o)|^2 \tag{5.183}$$

where w_o is the weight of the orientation in the quadrature.

The implementation has been designed to compute the absorption of circularly-polarized light on systems under the effect of an additional external magnetic field, B, which modifies the states $\tilde{P}$ and $\tilde{Q}$ for each orientation o through a Zeeman perturbation. The computed results are presented as the difference in the absorption of the left (-) and right (+) circularly-polarized light ($\Delta \mathrm{f}_{osc}$) and as the sum of the oscillator strength (f_{osc}), which corresponds to the linearly-polarized light absorption. The molecular orientations are constructed by using rotation matrices with three Euler angles: χ, θ, and ϕ. Herein, χ (the rotation angle on a plane perpendicular to the direction of external magnetic field/incident light) is integrated analytically whereas θ and ϕ are taken on a grid.

Finally, the states $\tilde{P}$ and $\tilde{Q}$ are obtained from QDPT by expanding the states over non-relativistic eigenstates of $\hat{H}_0$ ($\{I, J\}$) and the coefficients of the expansion ($d_{I\tilde{Q}}$) are obtained from the diagonalization of the complex matrix, which contains $\hat{H}_0$ as well as the SOC and Zeeman contributions (and SSC, if it is implemented in the selected electronic structure theory) expressed in $\{I, J\}$.

$$\langle \Psi_I^{SM} | \hat{H}_0 + \hat{H}_{SOC} + \hat{H}_{Zeeman} | \Psi_J^{S'M'} \rangle = \delta_{IJ}\delta_{SS'}\delta_{MM'}E_I^S + \langle \Psi_I^{SM} | \hat{H}_{SOC} + \hat{H}_{Zeeman} | \Psi_J^{S'M'} \rangle \quad (5.184)$$

5.33.2 Running and analyzing MCD calculations in TDDFT module

A minimum input to compute the Magnetic Circular Dichroism (MCD) requires setting the keyword DoMCD to true and to include an intensity for the external magnetic field B (in Gauss). An example input for the TD-DFT module is as follows:

```
! B3LYP def2-QZVPP

%tddft
    TDA False
    NROOTS 10
    DoSOC True
    DoMCD True
    B 50000.0
end

*xyz 0 1
 O     0.24287127056830      0.00033994295362      0.29479344369591
 C     0.13113617562040      0.00013705972874      1.65093880501262
 C     1.35780003491446     -0.00017874911331      2.22574232397512
 C     2.30247996517358     -0.00017814489451      1.14947275232337
 C     1.57300244613625      0.00013652073221      0.00793218434497
 H    -0.86951534119903      0.00025858475406      2.04160438459058
 H     1.56899490845286     -0.00038874637444      3.28047030533440
 H     3.37570289422870     -0.00038632501973      1.22171814100713
 H     1.83015664610447      0.00025985723335     -1.03506234028415
*
```

In the output file of this job, the estimated oscillator strengths (f_{osc}) and the difference between left and right circularly polarized light absorption ($\Delta \mathrm{f}_{osc}$) are provided:

```
-----------------------------------------------------------------------------
           MCD Transitions via transition electric dipole moments
                B =  50000.00 Gauss  T =   300.000 K
-----------------------------------------------------------------------------

                    dfosc             fosc           dfosc/fosc

 0 -> 1       -0.0000000000       0.0000000003      -0.0663180503
 0 -> 2        0.0000000000       0.0000000005       0.0000209714
 0 -> 3        0.0000000000       0.0000000003       0.0661609417
 0 -> 4       -0.0000000001       0.0000000004      -0.2991916107
 0 -> 5       -0.0000000000       0.0000000006      -0.0009270335
```

(continues on next page)

(continued from previous page)

```
 0 -> 6        0.0000000001       0.0000000004       0.2984154405
 0 -> 7        0.0000000003       0.0000000008       0.3721777693
 0 -> 8       -0.0000000000       0.0000000009      -0.0002960328
 0 -> 9       -0.0000000003       0.0000000008      -0.3689867758
 0 ->10       -0.0000050986       0.1741092913      -0.0000292840
```

These results may not be accurate when the energetic order of the states changes with respect to the relative orientation between the molecule and the external magnetic field. To obtain accurate results, it is necessary to perform a post-processing step for all orientations using the `orca_mapspc` program, which saves the results in a file that has the `.cis-el.dipole-length.1.mcd` extension:

```
orca_mapspc fur-mcd.cis-el.exact.1.mcd MCD -x050000 -x155000 -n10000 -w2000
```

In this example, we generate the spectrum (M^{-1}) between 50000 and 55000 cm^{-1} (-x050000 -x155000), using 10000 points (-n10000) and including a broadened normalized Gaussian function with a full width at half maximum of 2000cm^{-1} (-w2000).

Multiple MCD calculations can be performed in one run by setting multiple values for B. Transition moments can be also obtained through ED velocity formulation and FFMIO operator by setting the keywords `DoDipoleVelocity` and `DoFullSemiClassical` to true, respectively:

```
! B3LYP def2-QZVPP

%tddft
    TDA False
    nroots 10
    DoSOC True
    DoMCD True
    DoDipoleVelocity True
    DoFullSemiClassical True
    B 50000.0, 0.0
end
```

The results are printed separately in the output file for each setting:

```
-----------------------------------------------------------------------------
           MCD Transitions via transition electric dipole moments
                B =  50000.00 Gauss  T =   300.000 K
-----------------------------------------------------------------------------

                   dfosc              fosc            dfosc/fosc

 0 -> 1       -0.0000000000       0.0000000003      -0.0663180503
 .
 .
 .
 ------------------------------------------------------------------------------
           MCD Transitions via transition velocity dipole moments
                B =  50000.00 Gauss  T =   300.000 K
-----------------------------------------------------------------------------

                   dfosc              fosc            dfosc/fosc

 0 -> 1       -0.0000000001       0.0000000013      -0.0477214281
.
.
.

-----------------------------------------------------------------------------
           MCD Transitions via full Semi-classical formulation
                B =  50000.00 Gauss  T =   300.000 K
```

(continues on next page)

(continued from previous page)

```
-------------------------------------------------------------------------------

                    dfosc              fosc            dfosc/fosc

 0 -> 1       -0.0000000001       0.0000000016      -0.0731029604
.
.
.
```

Post-processing results are saved in the files having the `.cis-el.dipole-length.1.mcd`, `.cis-el.dipole-vel.1.mcd`, and `.cis-el.exact.1.mcd` extensions.

NOTE: It is worth enphasizing that the computed values of Δf_{osc} correspond to the difference in absorption between left and right circularly polarized light for the selected transition moments. In the case of both ED approximations, Δf_{osc} corresponds to the MCD signal. The sum of natural circular dichroism and magnetic-induced circular dichroism is obtained when the FFMIO is requested. To obtain only the MCD spectrum in an FFMIO scheme, it is necessary to subtract the natural circular dichroism by setting B to 0.0.

5.33.3 Running MCD calculations in other modules

The MCD implementation can also be used in other modules such as STEOM-CCSD, CAS, ROCIS, and MRCI (see the input file examples given below) by using the same keywords as those described for the TDDFT module.

```
%mrci
     DoDipoleLength true
     DoDipoleVelocity true
     DoFullSemiClassical true
     Soc
          DoSoc true
          DoMCD true
          B 50000.0
     end
end
```

```
%casscf
     DoDipoleLength true
     DoDipoleVelocity true
     DoFullSemiClassical true
     rel
          DoSoc true
          DoMCD true
          B 50000.0
     end
end
```

```
%rocis
  DoDipoleLength true
  DoDipoleVelocity true
  DoFullSemiClassical true
  rel
   DoSOC true
   DoMCD true
   B 50000.0
  end
end
```

```
%mdci
     DoSOC true
```

(continues on next page)

(continued from previous page)

```
    DoMCD true
    DoDipoleLength true
    DoDipoleVelocity true
    B 50000.0
end
```

It is important to keep in mind that the calculation of the MCD relies on the proper description of the transition moments and angular momentum for calculating the Zeeman perturbation. Therefore, the user is responsible for selecting the proper electronic structure method.

5.33.4 List of related keywords

```
%selected module
  DoMCD false          # Enables the use of the MCD module
  DoDipoleVelocity true # Use the electric dipolar velocity formulation
                        # for the light-matter interaction
  DoFullSemiClassical false  # Use the full semiclasical ligth-matter
                             # interaction transition moments
  MCDGridtype 1        # Grid for the molecular orientational average
                             # 1 = Lebedev grid (default)
                             # 2 = Regular grid
  MCDLebedev 14        # Number of points if Lebedev grid is selected
                       # 6, 12, 14(default), 26, 50, 110, 194, 302, 434, 590, 770
  NPOINTSPHI 10        # Number of points for phi angle if regular grid is selected
  NPOINTSTHETA 10      # Number of points for theta angle if regular grid is␣
↪selected
  B 3000.0             # Magnetic field in Gauss
  Temperature 300.0    # Self-explanatory. One temperature must be defined
                       # for each B value
end
```

5.34 Interface to SINGLE_ANISO Module

5.34.1 General Description

The `SINGLE_ANISO` program allows the non-perturbative calculation of effective spin (pseudospin) Hamiltonians and static magnetic properties of mononuclear complexes and fragments on the basis of an *ab initio*, including the spin-orbit interaction. As a starting point it uses the results of a `CASSCF/NEVPT2/SOC` calculation for the ground and several excited spin-orbit multiplets.

The following quantities can be computed:

- Parameters of pseudospin magnetic Hamiltonians (the methodology is described in [750]) :
  1. First order (linear after pseudospin) Zeeman splitting tensor (g tensor), including the determination of the sign of the product $g_X \cdot g_Y \cdot g_Z$.
  2. Second order (bilinear after pseudospin) zero-field splitting tensor (D tensor).
  3. Higher order zero-field splitting tensors $(D^2, D^4, D^6, ..., etc.)$
  4. Higher order Zeeman splitting tensors $(G^1, G^3, G^5, ..., etc.)$
- Crystal-Field parameters for the ground atomic $\tilde{J}$ multiplet for lanthanides. [751, 752]
- Crystal-Field parameters for the ground atomic $\tilde{L}$ term for lanthanides and transition metals.
- Static magnetic properties [753, 754]:
  1. Van Vleck susceptibility tensor $\chi_{\alpha\beta}(T)$.

2. Powder magnetic susceptibility function $\chi(T)$.
3. Magnetisation vector $\vec{M}(\vec{H})$ for specified directions of the applied magnetic field $\vec{H}$.
4. Powder magnetisation $\overline{M_{mol}}(H,T)$.
5. Magnetisation torque function $\vec{\tau}_{mol}(H,T)$.

The magnetic Hamiltonians are defined for a desired group of N low-lying electronic states obtained in CASSCF/SOC calculation to which a *pseudospin* $\tilde{S}$ is subscribed according to the relation $N = 2\tilde{S} + 1$. The *pseudospin* $\tilde{S}$ reduces to a true spin S in the absence of spin-orbit coupling. For instance, the two wave functions of a Kramers doublet correspond to the pseudospin $\tilde{S} = 1/2$. The implementation is done for any dimension of the pseudospin $\tilde{S}$, and controlled by the keyword MLTP.

The calculation of magnetic properties takes into account the contribution of excited states (the ligand-field and charge transfer states of the complex or mononuclear fragment which were included in the CASSCF/CASPT2 calculation) via their thermal population and Zeeman admixture. The effect of intermolecular exchange interaction between magnetic molecules on the resulting magnetic properties in a crystal is described by a phenomenological parameter zJ specified by the user.

5.34.2 Running SINGLE_ANISO Calculations

The SINGLE_ANISO is, in principle, a stand-alone utility (otool_single_aniso) that can be called directly from the shell with its own input file, provided that the *ab initio* datafile is available:

```
bash:$
bash:$  $ORCA/x86_64/otool_single_aniso < single_aniso.input > single_aniso.output
bash:$
```

However, this usage may not be so convenient, as the file single_aniso.input must include the true name of the datafile. For the user's convenience, a deeper integration between SINGLE_ANISO and CASSCF program in ORCA was implemented, as described below.

As a prerequisite for using the SINGLE_ANISO module to calculate the magnetic properties of the investigated compound, **spin-orbit coupling and other relativistic effects are already taken fully into account at the stage of quantum chemistry calculation of the investigated compound**. The necessary information of the *ab initio* calculation is provided in a form of a "datafile": energy spectra, angular momentum integrals, etc. The interface with ORCA generates the required datafile automatically. The following naming conventions were adopted for the datafile in function of the employed computational method:

- CASSCF+SOC+SINGLE_ANISO => "$orca_input_name.CASSCF.anisofile"
- CASSCF+QD-NEVPT2+SOC+SINGLE_ANISO => "$orca_input_name.NEVPT2.anisofile"

Note that if the CASSCF+QD-NEVPT2+SOC+SINGLE_ANISO calculation is requested, then the SINGLE_ANISO will be executed twice, and the above two datafiles will be generated. The interface will generate the SINGLE_ANISO input file with the keywords information provided in the CASSCF/aniso subblock. These filename of the datafile is included automatically in the input file for the SINGLE_ANISO utility (keyword DATA), also generated automatically by the interface. The naming convention for the generated input files for the SINGLE_ANISO utility is "$orca_input_name.anisofile".

All keywords of the SINGLE_ANISO program are possible to be specified within the CASSCF/ANISO subblock. They are referenced in Section *Reference List of CASSCF/ANISO Keywords*. Optionally, a working SINGLE_ANISO input file can be passed directly to the CASSCF module setting the filename with keyword InputNameOnDisk in the ANISO subblock.

An example of the full ORCA input for performing magnetic properties calculations within the CASSCF/SOC/SINGLE_ANISO methodology for a hypothetical Co(II) compound is provided below:

```
! 6-31G TightSCF   # basis set and other global ORCA settings

%maxcore 2000
```

(continues on next page)

(continued from previous page)

```
%casscf nel 7
  norb 5           # 7 electrons in 5 orbitals (3d shell)
  mult 4, 2
  nroots 10, 40     # 10 quartet and 40 doublet states

  rel
  dosoc true       # include spin-orbit coupling
  end

  ANISO
    doaniso true     # generate datafile/input and call
                     # the SINGLE_ANISO module
    MLTP 2,2,2,2     # group of spin-orbit states for which the pseudospin
                     # Hamiltonian is computed: 4 low-lying spin-orbit doublet␣
↪states.

    TINT 0, 300, 301 # 301 steps in the temperature interval [0-300]
                     # for magnetic susceptibility (in Kelvin)
    HINT 0, 7.0, 71  # 71 steps in the field interval [0-7]
                     # for molar magnetisation (in Tesla)
    TMAG 1.0,1.2,1.8,2.5,3.6  # temperature points for which molar magnetisation
                                            # is computed
     CRYS_element "Co"
     CRYS_charge 2
     PLOT true         # requires the ANISO to produce gnuplot scripts,
                              # datafiles and plots of various quantities

    # Alternative to the snippet above. Provide separate input file:
    # InputFile "$orca_input_name.anisoinput"
  end
end

*xyz 0 4   # charge is 0 for this neutral compound
Co   -2.80118000    9.91634000   19.40386000
O    -3.59660000   12.00284000   20.51731000
O    -5.12835000   10.85934000   19.53431000
O    -5.70975000   12.39302000   20.99406000
O    -1.30202341   11.67611386   19.17300658
O    -3.84191000    9.45315000   21.48634000
O    -1.27500262    8.12582233   19.18634310
O    -3.94611990    9.65426823   17.48476360
N    -4.85020000   11.78071000   20.36823000
H    -1.23636310   12.09677337   18.41017549
H    -1.07910455    7.59540828   19.85227241
H    -3.30514987    9.28034259   22.26327382
H    -4.79957696    9.43862752   21.55163236
H    -4.64801074    9.00163025   17.42987361
H    -3.73273676   10.19508893   16.72083912
H    -0.75470916   11.94100908   19.91589125
*
```

The input above utilises the following keywords: `MLTP` keyword requires the computation of the g tensor for 4 groups of spin-orbit states, the dimensionality of each group being 2 (Kramers or Ising doublets). `TINT` requires computation of the magnetic susceptibility in the temperature interval 0 K - 300 K distributed equally in 300 temperature intervals. `TMAG` requires computation of powder molar magnetisation at 6 temperature points, in Kelvin (K): 1.0 K, 1.2 K, 1.8 K, 2.5 K, 2.9 K and 3.6 K. `HINT` defines the range for the magnetic field strength, in Tesla. `PLOT` keyword invokes the plotting function of the module. `CRYS_element + CRYS_charge` request for the computation of the crystal field parameters for the ground term of the Co^{2+} ion. For more information about the keywords in `SINGLE_ANISO` module, you can refer to section *Reference List of CASSCF/ANISO Keywords*.

Please always check the obtained orbitals after `CASSCF` calculation. In this particular case, the active orbitals

(45-49) are localised on the Co site and display dominant $3d$ character.

```
                 45        46        47        48        49
           -0.37264  -0.36672  -0.36520  -0.36153  -0.35018
            1.40513   1.40270   1.39998   1.39823   1.39395
            --------  --------  --------  --------  --------
0 Co s         0.0       0.0       0.0       0.0       0.2
0 Co pz        0.1       0.1       0.1       0.0       0.0
0 Co px        0.0       0.2       0.0       0.0       0.1
0 Co py        0.1       0.1       0.0       0.0       0.1
0 Co dz2      23.6      39.5      10.0       2.0      23.5
0 Co dxz      12.8      30.8      48.6       4.4       2.2
0 Co dyz      46.5      10.4      19.3      20.8       2.4
0 Co dx2y2     6.9       0.3      21.2      69.2       1.6
0 Co dxy       9.5      17.6       0.2       2.8      68.0
```

We see that in the above output section, the five active orbitals have dominant contribution from the Co-$3d$ basis functions. This is OK and is expected for common transition metal compounds. For lanthanide compounds, the seven active orbitals should have dominant contribution from the $4f$ shell. Larger active spaces must be carefully inspected and analysed. We refer here to the respective section of this manual describing the CASSCF method and how to achieve convergence *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)*.

The results calculated by using `SINGLE_ANISO` module are placed after the `SOC` section in ORCA output. Here is the explanation for these results.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CALCULATION OF PSEUDOSPIN HAMILTONIAN TENSORS FOR THE MULTIPLET 1 ( effective S = ␣
↪1/2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
The pseudospin is defined in the basis of the following spin-orbit states:
spin-orbit state 1. energy(1) =       0.000 cm-1.
spin-orbit state 2. energy(2) =       0.000 cm-1.

g TENSOR:
|-------------------------------------------------------|
|    MAIN VALUES    |             MAIN MAGNETIC AXES     |  x , y , z  -- initial␣
↪Cartesian axes
|-------------------|----|----- x ------- y ------- z ---|  Xm, Ym, Zm -- main␣
↪magnetic axes
| gX =   0.09871069 | Xm | -0.456536 -0.363638  0.811998 |
| gY =   0.11729280 | Ym |  0.643532  0.495246  0.583605 |
| gZ =  11.21949040 | Zm | -0.614361  0.788985  0.007914 |
|-------------------------------------------------------|
The sign of the product gX * gY * gZ for multiplet 1: < 0.
```

The section above shows the g tensor for the ground Kramers doublet. Since the g_X and g_Y are much smaller than the g_Z component, the Zm axis is denoted as the $mainmagneticaxis$ of the computed molecule. The "Zm | -0.614361 0.788985 0.007914 |" denotes the Cartesian components of the Zm vector.

In the case the computation of the parameters of the crystal field is requested by `CRYS_element` and `CRYS_charge`, the following section will be found in the output:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     CALCULATION OF CRYSTAL-FIELD PARAMETERS OF THE GROUND ATOMIC TERM, L =  3.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

The parameters of the Crystal Field matrix are written in the coordinate system:
(Xm, Ym, Zm) --  the main magnetic axes of the ground pseudo-L = | 3> orbital␣
↪multiplet.
Rotation matrix from the initial coordinate system to the employed coordinate␣
↪system is:
-------------------------------------------------------------------|
x , y , z  -- initial Cartesian axes                                |
```

(continues on next page)

(continued from previous page)

```
Xm, Ym, Zm -- main magnetic axes                                    |
                     x                 y                  z         |
      | Xm | -0.61155332461133 -0.79120321735748  0.00000000001900 |
 R =  | Ym |  0.79120321735748 -0.61155332461133 -0.00000000000264 |
      | Zm |  0.00000000001371  0.00000000001342  1.00000000000000 |
-------------------------------------------------------------------|
Quantization axis is Zm.


-------------------------------------------------------------------------------
↪-----------|
   Ab Initio Crystal-Field Splitting Matrix written in the basis of Pseudo-L␣
↪Eigenfunctions   |
-------------------------------------------------------------------------------
↪-----------|
          |                    | -3 >               |                    | -2 >   ␣
↪           |
----------|---------- REAL ----------- IMAGINARY ---|---------- REAL -----------␣
↪IMAGINARY ---|
   < -3 | |  -777.2218617165776     0.0000000000000 |    -0.0000000454613     0.
↪0000000351393 |
   < -2 | |    -0.0000000454613    -0.0000000351393 |   285.4563720817839     0.
↪0000000000000 |
   < -1 | |     0.0285952366255    -0.0055088548681 |     0.0000000002580    -0.
↪0000000024089 |
   <  0 | |    -0.0000000160911     0.0000000604843 |     0.0001933353720    -0.
↪0013498718536 |
   <  1 | |     0.0116130821579    -0.0096419531309 |    -0.0000000005651    -0.
↪0000000026613 |
   <  2 | |     0.0000000070279    -0.0000000332291 |     0.0141002831881    -0.
↪0117070160056 |
   <  3 | |    -0.0000002782243     0.0000001745761 |    -0.0000000070278     0.
↪0000000332291 |
-------------------------------------------------------------------------------
↪-----------|
-------------------------------------------------------------------------------
↪-----------|
          |                    | -1 >               |                    |  0 >   ␣
↪           |
----------|---------- REAL ----------- IMAGINARY ---|---------- REAL -----------␣
↪IMAGINARY ---|
   < -3 | |     0.0285952366255     0.0055088548681 |    -0.0000000160911    -0.
↪0000000604843 |
   < -2 | |     0.0000000002580     0.0000000024089 |     0.0001933353720     0.
↪0013498718536 |
   < -1 | |   347.8121781289439    -0.0000000000000 |     0.0000000069613    -0.
↪0000000861891 |
   <  0 | |     0.0000000069613     0.0000000861891 |   287.9066230116981     0.
↪0000000000000 |
   <  1 | |    -0.0106576508828     0.0003303251251 |    -0.0000000069613    -0.
↪0000000861891 |
   <  2 | |     0.0000000005651     0.0000000026613 |     0.0001933353720    -0.
↪0013498718536 |
   <  3 | |     0.0116130821579    -0.0096419531309 |     0.0000000160911    -0.
↪0000000604843 |
-------------------------------------------------------------------------------
↪-----------|
-------------------------------------------------------------------------------
↪-----------|
          |                    |  1 >               |                    |  2 >   ␣
↪           |
```

(continues on next page)

(continued from previous page)

```
----------|---------- REAL ----------- IMAGINARY ---|---------- REAL -----------␣
↪IMAGINARY ---|
   < -3 | |     0.0116130821579     0.0096419531309 |     0.0000000070279     0.
↪0000000332291 |
   < -2 | |    -0.0000000005651     0.0000000026613 |     0.0141002831881     0.
↪0117070160056 |
   < -1 | |    -0.0106576508828    -0.0003303251251 |     0.0000000005651    -0.
↪0000000026613 |
   <  0 | |    -0.0000000069613     0.0000000861891 |     0.0001933353720     0.
↪0013498718536 |
   <  1 | |   347.8121781289439    -0.0000000000000 |    -0.0000000002580    -0.
↪0000000024089 |
   <  2 | |    -0.0000000002580     0.0000000024089 |   285.4563720817837     0.
↪0000000000000 |
   <  3 | |     0.0285952366255    -0.0005088548681 |     0.0000000454613     0.
↪0000000351392 |
-------------------------------------------------------------------------------
↪-----------|
-----------------------------------------------------|
          |                     |  3 >               |
----------|---------- REAL ----------- IMAGINARY ---|
   < -3 | |    -0.0000002782241    -0.0000001745761 |
   < -2 | |    -0.0000000070278    -0.0000000332291 |
   < -1 | |     0.0116130821579     0.0096419531309 |
   <  0 | |     0.0000000160911     0.0000000604843 |
   <  1 | |     0.0285952366255     0.0005088548681 |
   <  2 | |     0.0000000454613    -0.0000000351392 |
   <  3 | |  -777.2218617165773     0.0000000000000 |
-----------------------------------------------------|
```

In the above section, the low-lying CASSCF states of the Co^{2+} site originating from the free ion ^{4}F term are transformed towards the eigenstates of the ($\tilde{L} = 3$), and the low-lying CASSCF *diagonal* 7×7 energy matrix is re-written in this basis. The *non-diagonal* "Ab Initio Crystal-Field Splitting Matrix" is printed in the above section. In the subsequent output sections, the obtained crystal field matrix is decomposed in a linear combination of Irreducible Tensorial Operators (ITOs) and the obtained expansion coefficients are listed in the output.

$$\hat{H}_{CF} = \sum_{k}^{2L} \sum_{q}^{-k,+k} B_k^q \hat{O}_k^q(\tilde{L})$$

The parameters are given for several sets of ITOs.

```
*****************************************************************************
The Crystal-Field Hamiltonian:
   Hcf = SUM_{k,q} * [ B(k,q) * O(k,q) ];
where:
   O(k,q) =  Extended Stevens Operators (ESO)as defined in:
          1. Rudowicz, C.; J.Phys.C: Solid State Phys.,18(1985) 1415-1430.
          2. Implemented in the "EasySpin" function in MATLAB, www.easyspin.org.
   k - the rank of the ITO, = 2, 4, 6, 8, 10, 12.
   q - the component of the ITO, = -k, -k+1, ... 0, 1, ... k;
Knm are proportionality coefficients between the ESO and operators defined in
J. Chem. Phys., 137, 064112 (2012).
-------------------------------------------------|
  k |  q  |    (K)^2    |         B(k,q)         |
----|-----|-------------|------------------------|
  2 |  -2 |       1.50  |  0.44029016547734E-03  |
  2 |  -1 |       6.00  |  0.24547763681975E-08  |
  2 |   0 |       1.00  | -0.43693326103120E+02  |
  2 |   1 |       6.00  | -0.84162317914775E-08  |
  2 |   2 |       1.50  |  0.12672200639220E-02  |
----|-----|-------------|------------------------|
```

(continues on next page)

(continued from previous page)

```
  4 |  -4 |      17.50  |  0.20185049671189E-03 |
  4 |  -3 |     140.00  | -0.24080325997038E-08 |
  4 |  -2 |      10.00  |  0.53646717565242E-04 |
  4 |  -1 |      20.00  |  0.99248109880376E-09 |
  4 |   0 |       1.00  | -0.67496280141952E+00 |
  4 |   1 |      20.00  | -0.53708624488205E-09 |
  4 |   2 |      10.00  |  0.33333801678482E-03 |
  4 |   3 |     140.00  | -0.66502585057483E-09 |
  4 |   4 |      17.50  |  0.24311509626065E-03 |
----|-----|-------------|-----------------------|
  6 |  -6 |      57.75  | -0.48493356946616E-09 |
  6 |  -5 |     693.00  |  0.45219078679397E-09 |
  6 |  -4 |      31.50  |  0.11222605476277E-05 |
  6 |  -3 |     105.00  | -0.14936428088413E-09 |
  6 |  -2 |      26.25  |  0.68538037767693E-06 |
  6 |  -1 |      42.00  | -0.24067665895440E-09 |
  6 |   0 |       1.00  | -0.18259217459128E-02 |
  6 |   1 |      42.00  |  0.11394843516111E-11 |
  6 |   2 |      26.25  |  0.48314159464149E-05 |
  6 |   3 |     105.00  | -0.33636623245296E-10 |
  6 |   4 |      31.50  |  0.13517294099051E-05 |
  6 |   5 |     693.00  |  0.95637007025194E-10 |
  6 |   6 |      57.75  | -0.77284500243776E-09 |
-------------------------------------------------|
```

In the sections below, the weight of various expansion terms on the total energy splitting of the corresponding term or multiplet is analysed.

```
CUMULATIVE WEIGHT OF INDIVIDUAL-RANK OPERATORS ON THE CRYSTAL FIELD SPLITTING:
  O2 :-----------------------------------------:  70.094642 %.
  O2 + O4 :--------------------------------------:  99.417093 %.
  O2 + O4 + O6 :---------------------------------: 100.000000 %.

ENERGY SPLITTING INDUCED BY CUMULATIVE INDIVIDUAL-RANK OPERATORS (in cm-1).
----------|---------------|---------------|---------------|---------------|
  L = 3   |     RASSCF    |     ONLY      |     ONLY      |     ONLY      |
          |     INITIAL   |      O2       |     O2+O4     |   O2+O4+O6    |
----------|---------------|---------------|---------------|---------------|
 w.f.  1  |    0.00000000 |    0.00000000 |    0.00000000 |    0.00000000 |
 w.f.  2  |    0.00000220 |    0.00000000 |    0.00000149 |    0.00000220 |
 w.f.  3  | 1062.65990786 |  655.39989137 | 1058.22649805 | 1062.65990786 |
 w.f.  4  | 1062.69656233 |  655.39989157 | 1060.35861560 | 1062.69656233 |
 w.f.  5  | 1065.12848829 | 1048.63177735 | 1060.39653667 | 1065.12848829 |
 w.f.  6  | 1125.02338078 | 1048.64787570 | 1129.62295551 | 1125.02338078 |
 w.f.  7  | 1125.04470493 | 1179.71980502 | 1129.64777494 | 1125.04470493 |
----------|---------------|---------------|---------------|---------------|


WEIGHT OF INDIVIDUAL-RANK OPERATORS ON THE CRYSTAL FIELD SPLITTING:
  O2  :-----------------------------------------:  70.094642 %.
  O4  :-----------------------------------------:  29.322451 %.
  O6  :-----------------------------------------:   0.582907 %.


ENERGY SPLITTING INDUCED BY INDIVIDUAL-RANK OPERATORS (in cm-1).
----------|---------------|---------------|---------------|---------------|
  L = 3   |     RASSCF    |     ONLY      |     ONLY      |     ONLY      |
          |     INITIAL   |      O2       |      O4       |      O6       |
----------|---------------|---------------|---------------|---------------|
 w.f.  1  |    0.00000000 |    0.00000000 |    0.00000000 |    0.00000000 |
 w.f.  2  |    0.00000220 |    0.00000000 |  121.49328633 |    0.00351330 |
```

(continues on next page)

(continued from previous page)

```
 w.f.   3   | 1062.65990786 |  655.39989137 |  121.49330341 |    4.60307936 |
 w.f.   4   | 1062.69656233 |  655.39989157 |  202.46860036 |    4.60307986 |
 w.f.   5   | 1065.12848829 | 1048.63177735 |  202.50909974 |    6.90310777 |
 w.f.   6   | 1125.02338078 | 1048.64787570 |  526.45202593 |    6.90437338 |
 w.f.   7   | 1125.04470493 | 1179.71980502 |  526.48994463 |   11.50506445 |
----------|---------------|---------------|---------------|---------------|


WEIGHT OF INDIVIDUAL CRYSTAL FIELD PARAMETERS ON THE CRYSTAL FIELD SPLITTING: (in␣
↪descending order):
CFP are given in ITO used in J. Chem. Phys. 137, 064112 (2012).
-------------------------------------------------------|
  k |  q  |         B(k,q)         |    Weight (in %)    |
----|-----|------------------------|---------------------|
2 |   0 | -0.43693326103120E+02 |  70.08577012260780 |
4 |   0 | -0.67496280141952E+00 |  29.31873954450217 |
6 |   0 | -0.18259217459128E-02 |   0.58283348856981 |
4 |   2 |  0.10541073637635E-03 |   0.00457878555457 |
4 |   4 |  0.58115623682363E-04 |   0.00252440109385 |
4 |  -4 |  0.48251497695683E-04 |   0.00209592749496 |
2 |   2 |  0.10346808494753E-02 |   0.00165966774870 |
4 |  -2 |  0.16964581649793E-04 |   0.00073690009261 |
2 |  -2 |  0.35949541472835E-03 |   0.00057664442705 |
6 |   2 |  0.94299583491020E-06 |   0.00030100389209 |
6 |   4 |  0.24084325388052E-06 |   0.00007687706999 |
6 |  -4 |  0.19995783180553E-06 |   0.00006382645967 |
6 |  -2 |  0.13377255211448E-06 |   0.00004270014495 |
6 |   6 | -0.10169893585902E-09 |   0.00000003246226 |
6 |  -6 | -0.63812572794630E-10 |   0.00000002036895 |
6 |  -1 | -0.37137214734348E-10 |   0.00000001185418 |
4 |  -1 |  0.22192552033089E-09 |   0.00000000963990 |
4 |  -3 | -0.20351589971646E-09 |   0.00000000884023 |
2 |   1 | -0.34359122410183E-08 |   0.00000000551133 |
6 |  -5 |  0.17177307577593E-10 |   0.00000000548299 |
4 |   1 | -0.12009613533364E-09 |   0.00000000521668 |
6 |  -3 | -0.14576461261073E-10 |   0.00000000465280 |
4 |   3 | -0.56204942711777E-10 |   0.00000000244141 |
2 |  -1 |  0.10021582557877E-08 |   0.00000000160750 |
6 |   5 |  0.36329494838219E-11 |   0.00000000115964 |
6 |   3 | -0.32825983078827E-11 |   0.00000000104780 |
6 |   1 |  0.17582625268298E-12 |   0.00000000005612 |
----|-----|------------------------|---------------------|
```

In the case of lanthanide compounds, the same keywords (`CRYS_element` and `CRYS_charge`) trigger the energy decomposition of the lowest energy matrix corresponding to the ground $J-$ multiplet of the respective lanthanide ion.

```
%%%
```

(continued from previous page)

```
|     |            |  SUM (Z)     |     (zJ=0)    |              |              |      ␣
↪      |
|-----|-------------------------------------------------------------------------------
↪------|
|Units|   Kelvin   |     ---      |  cm3*K/mol   |  cm3*K/mol   |   cm3/mol    |   mol/
↪cm3   |
|-----|-------------------------------------------------------------------------------
↪------|
|     |   0.000100 | 0.20000E+01 |  3.93592010 |  3.93592010 | 0.39359E+05 | 0.
↪25407E-04 |
|     |   1.000000 | 0.20000E+01 |  3.94046247 |  3.94046247 | 0.39405E+01 | 0.
↪25378E+00 |
|     |   2.000000 | 0.20000E+01 |  3.94500530 |  3.94500530 | 0.19725E+01 | 0.
↪50697E+00 |
|     |   3.000000 | 0.20000E+01 |  3.94954814 |  3.94954814 | 0.13165E+01 | 0.
↪75958E+00 |
|     |   4.000000 | 0.20000E+01 |  3.95409097 |  3.95409097 | 0.98852E+00 | 0.
↪10116E+01 |
|     |   5.000000 | 0.20000E+01 |  3.95863380 |  3.95863380 | 0.79173E+00 | 0.
↪12631E+01 |
|     |   6.000000 | 0.20000E+01 |  3.96317663 |  3.96317663 | 0.66053E+00 | 0.
↪15139E+01 |
|     |   7.000000 | 0.20000E+01 |  3.96771946 |  3.96771946 | 0.56682E+00 | 0.
↪17642E+01 |
|     |   8.000000 | 0.20000E+01 |  3.97226229 |  3.97226229 | 0.49653E+00 | 0.
↪20140E+01 |
|     |   9.000000 | 0.20000E+01 |  3.97680512 |  3.97680512 | 0.44187E+00 | 0.
↪22631E+01 |
|     |  10.000000 | 0.20000E+01 |  3.98134795 |  3.98134795 | 0.39813E+00 | 0.
↪25117E+01 |
...
```

This section shows the computed magnetic susceptibility. The formula used for this calculation assumes the zero-field limit, $i.e. H = 0.0$ Tesla. A picture called "XT_no_field.png" using the above data will be created in the working directory whenever the `PLOT` keyword is included in the `SINGLE_ANISO` input. The picture shows the temperature dependence of the magnetic susceptibility.

```
|-------------------------------------------------------------------------------
↪--------------------|
|                      VAN VLECK SUSCEPTIBILITY TENSOR FOR zJ = 0,  in cm3*K/mol  ␣
↪                    |
|-------------------------------------------------------------------------------
↪--------------------|
|   T(K)    | |         SUSCEPTIBILITY TENSOR      |  MAIN VALUES  |              ␣
↪MAIN AXES             |
|----------|-|----- x --------- y --------- z ---|---------------|------ x --------
↪- y --------- z ----|
|          |x|  4.456611   -5.721848   -0.057261 | X:   0.000914 |  0.45654560  0.
↪36364537 -0.81199025 |
| 0.000100 |y| -5.721848    7.349367    0.073827 | Y:   0.001291 |  0.64352653  0.
↪49524178  0.58361733 |
|          |z| -0.057261    0.073827    0.001782 | Z:  11.805555 | -0.61436123  0.
↪78898519  0.00791499 |
|----------|-|----- x --------- y --------- z ---|---------------|------ x --------
↪- y --------- z ----|
|          |x|  4.461142   -5.718873   -0.057275 | X:   0.007129 |  0.48578721  0.
↪38619740 -0.78413160 |
| 1.000000 |y| -5.718873    7.351927    0.074460 | Y:   0.008275 |  0.62173382  0.
↪47788368  0.62054351 |
|          |z| -0.057275    0.074460    0.008319 | Z:  11.805983 | -0.61437598  0.
↪78897323  0.00796220 |
```

(continues on next page)

(continued from previous page)

```
|----------|-|----- x --------- y --------- z ---|---------------|------ x --------
↪- y --------- z ----|
|          |x|  4.465674   -5.715898   -0.057290 | X:   0.013344 | -0.49137357 -0.
↪39055217  0.77847352 |
| 2.000000 |y| -5.715898    7.354486    0.075093 | Y:   0.015261 |  0.61731357  0.
↪47435127  0.62762635 |
|          |z| -0.057290    0.075093    0.014856 | Z:  11.806411 | -0.61439073  0.
↪78896126  0.00800947 |
...
```

The section above shows how the main axes of the susceptibility tensor evolves with temperature.

```
        HIGH-FIELD POWDER MAGNETIZATION
              (Units: Bohr magneton)

|-----------|---------------|---------------|---------------|---------------|------
↪---------|
|    H(T)   |STATISTICAL SUM|     1.000 K.  |     1.200 K.  |     1.800 K.  |    ␣
↪2.500 K.  |
|-----------|---------------|---------------|---------------|---------------|------
↪---------|
|  0.000100 |     1.9995371 |  0.0007055560 |  0.0005880989 |  0.0003923371 |  0.
↪0002827105 |
|  0.100000 |     1.6293687 |  0.6863212310 |  0.5768343572 |  0.3889480349 |  0.
↪2814379254 |
|  0.200000 |     1.3961049 |  1.2730827904 |  1.0928431358 |  0.7585259147 |  0.
↪5554332924 |
|  0.300000 |     1.2492960 |  1.7176941099 |  1.5137449727 |  1.0936561795 |  0.
↪8154486026 |
|  0.400000 |     1.1568991 |  2.0312460704 |  1.8358752195 |  1.3858429640 |  1.
↪0565149122 |
|  0.500000 |     1.0987474 |  2.2456644189 |  2.0736324473 |  1.6329635867 |  1.
↪2755212118 |
|  0.600000 |     1.0621485 |  2.3917509695 |  2.2464760254 |  1.8374994655 |  1.
↪4711430994 |
|  0.700000 |     1.0391143 |  2.4924803644 |  2.3720174309 |  2.0044495622 |  1.
↪6435265950 |
|  0.800000 |     1.0246173 |  2.5633469179 |  2.4639356325 |  2.1396808073 |  1.
↪7938697856 |
|  0.900000 |     1.0154934 |  2.6144012337 |  2.5321303741 |  2.2489088878 |  1.
↪9240150206 |
|  1.000000 |     1.0097510 |  2.6521008670 |  2.5835380488 |  2.3371993536 |  2.
↪0361143785 |
|  1.100000 |     1.0061370 |  2.6806180412 |  2.6229602467 |  2.4088035375 |  2.
↪1323879997 |
|  1.200000 |     1.0038624 |  2.7026854246 |  2.6537186141 |  2.4671744880 |  2.
↪2149684436 |
|  1.300000 |     1.0024309 |  2.7201250266 |  2.6781249799 |  2.5150622492 |  2.
↪2858129074 |
|  1.400000 |     1.0015299 |  2.7341759016 |  2.6978046065 |  2.5546323752 |  2.
↪3466634357 |
...
```

This section shows the field dependence of the powder molar magnetisation. A picture named "MH.png" can be created by using the `PLOT` keyword in the `SINGLE_ANISO` input file.

Running `CASSCF` calculations on lanthanides compounds in ORCA might be a bit more cumbersome compared to transition metal compounds, due to the convergence of this method. However, following the instructions in the *Complete and Incomplete Active Space Self-Consistent Field (CASSCF and RAS/ORMAS)* section and the related tips in this manual and on the Forum, the calculations could be performed. From our experience, the main reason for the poor convergence of `CASSCF` calculation originates from the wrong orbitals occupying the active space. This issue can be overcame by performing a proper rotation of the molecular orbitals such that the seven orbitals with dominant

$4f$ contribution are placed in the active space. As soon as the active orbitals acquire the dominant $4f$ weight, the convergence is quite straightforward.

Below we describe the calculation on a lanthanide fragment [Ce(COT) $_2$]$^-$ (COT=(C_8H_8) $^{2-}$) as an example:

```
!DKH DKH-DEF2-SVP slowconv KDIIS BP

%basis
newgto Ce "SARC2-DKH-QZVP" end
end

%scf
MaxIter 500
end

*xyz -1 2
Ce   5.97600100    5.09133100   13.17268800
C    5.47882500    2.98632700   11.42941100
C    4.38424700    3.88677900   11.27367600
H    4.21867900    4.06431900   10.35453600
C    3.47138800    4.59373300   12.09958800
H    2.87027600    5.12178400   11.58692300
C    3.21937800    4.72005000   13.49499100
C    3.84198900    4.08874200   14.61728000
H    3.46926600    4.38472800   15.43857500
C    4.86395700    3.14327700   14.81900800
H    4.97094200    2.92197400   15.73690500
C    5.77247800    2.44085400   13.98883400
H    6.34594800    1.86846400   14.48498500
C    6.03182900    2.38537000   12.60408600
H    6.75450600    1.80396200   12.40022700
C    6.40698800    7.65195200   12.14877700
C    6.11546300    7.83965300   13.53689900
H    5.47247100    8.52635500   13.66635300
C    6.52698400    7.27593700   14.77461400
H    6.07344100    7.67704500   15.50605500
C    7.44425200    6.26569500   15.21837900
C    8.37896000    5.47470700   14.49350900
H    8.88315100    4.90771300   15.06566300
C    8.74883600    5.31701900   13.13468700
H    9.45277500    4.68873000   13.02529900
C    8.32602800    5.86701900   11.90372100
H    8.81295200    5.51473700   11.16784000
C    7.36115600    6.80638600   11.50204100
H    7.33662500    6.90506000   10.55697400
H    5.93270067    2.68976505   10.50694264
H    5.83417492    8.22959522   11.45371334
H    2.43475960    5.39234559   13.77300645
H    7.40021961    6.07954201   16.27114118
*
```

This is the first step of the calculation. For heavier elements like lanthanides, we must consider relativistic effect by using `DKH` keyword. We explicitly use `KDIIS` in the calculation to smoothen out convergence. The orbital file called "CeCOT2_1.gbw" will be generated after this step. We further use this gbw file to do the `CASSCF` calculation.

```
!DKH DKH-DEF2-SVP TightSCF conv Moread

%moinp "CeCOT2_1.gbw"

%basis
newgto Ce "SARC2-DKH-QZVP" end
end
```

(continues on next page)

(continued from previous page)

```
%casscf nel 1
   norb 7  # 1 electrons in 7 f orbitals
   mult 2
   nroots 7  # 7 doublet states

   rel
     dosoc true  # include spin-orbit coupling
   end
end

*xyz -1 2
......
*
```

We need to check the orbitals after the CASSCF step with the orbital file named "CeCOT2_2.gbw" obtained.

```
              85        86        87        88        89        90        91
           0.45543   0.45310   0.27655   0.45085   0.45251   0.45760   0.45713
           0.14286   0.14286   0.14286   0.14286   0.14286   0.14286   0.14286
           --------  --------  --------  --------  --------  --------  --------
0 Ce f0       26.2       0.9       0.0       1.3       3.3       0.5       6.6
0 Ce f+1       5.4      20.2       0.0      24.0       3.4       5.6      39.1
0 Ce f-1       0.4      30.4       0.0      52.8      10.7       0.6       3.9
0 Ce f+2       3.8       0.2       0.7       1.4       4.7      76.7      10.7
0 Ce f-2      50.7       0.7       0.0       2.1       4.2       4.0       1.3
0 Ce f+3       8.2      22.0       0.0       1.2      55.8       0.0      11.6
0 Ce f-3       4.7      25.3       0.0      16.5      17.1      10.1      25.2
```

Orbitals 85, 86, 88-91 and 130 are occupied and strongly metal based $4f$-orbitals. For comparison, the converged CASSCF orbitals are pure $4f$-orbitals (99% metal-based). The orbitals need to be rotated in order to fit the active space (85-91). Then we can use the results of CASSCF/SOC calculation to call for the SINGLE_ANISO program.

```
!DKH DKH-DEF2-SVP TightSCF conv Moread
%moinp "CeCOT2_2.gbw"

%basis
newgto Ce "SARC2-DKH-QZVP" end
end

%scf rotate {87,130,90} end
end

%casscf nel 1
   norb 7
   mult 2
   nroots 7

   rel
   dosoc true
   end

   ANISO
     doaniso true
     MLTP 2,2,2    # 3 Kramers doublets, J=5/2
     MAVE 1, 12   # nsym=1, Lebedev grid number 12
     XFIE 0.1        # the applied magnetic field is 0.1 T
     CRYS_element "Ce"
     CRYS_charge 3
     NCUT 14
     ABCC_abc     11.0735, 12.6738, 22.4854, 84.436, 86.690, 83.969
     ABCC_center  0.82682, 0.31234, 0.78619
```

(continues on next page)

(continued from previous page)

```
     ZJPR  -0.120
     HEXP_temp  2.0, 3.0
     HEXP_H  0.0, 1.0, 2.0, 3.0, 4.0
     HEXP_M[0]= 0.0, 2.46, 2.86, 2.95, 2.98
     HEXP_M[1]= 0.0, 2.04, 2.68, 2.87, 2.94
     TEXP_temp  0.0, 10.0, 20.0, 30.0, 40.0, 50.0
     TEXP_chiT  4.5, 4.5, 4.58, 4.62, 4.66, 4.70
     UBAR true
     PLOT true
     ZEEM[0]=1.0, 0.0, 0.0
     ZEEM[1]=0.0, 0.0, 1.0
     ZEEM[2]=0.0, 1.0, 0.0
     ZEEM[3]=0.75, 0.0, 0.25
   end
end

*xyz -1 2
......
*
```

The order of the keywords listed in the `CASSCF/ANISO` subblock does not matter.

5.34.3 Reference List of CASSCF/ANISO Keywords

The only required keyword for `SINGLE_ANISO` is the `DATA`, specifying the name of the datafile containing the *ab initio* information. The ORCA interface includes this keyword automatically and therefore it is not referenced here. All other keywords are extra and allow various customisation of the execution. For the computation of the EPR g-tensor, the only unknown variable for `SINGLE_ANISO` is the dimension (multiplicity) of the pseudospin(s). This information can be provided by the `MLTP` keyword. For example, in cases where spin-orbit coupling is weak, the multiplicity of the effective spin Hamiltonian is usually the same as the multiplicity of the lowest term (e.g. high spin Fe^{3+}: $S = \tilde{S} = 5/2$), while in the cases with strong anisotropy (lanthanide, actinide complexes, Co^{2+} complexes, cases with near-orbital degeneracy, etc.) the lowest energy levels form a group of states which may differ drastically from the spins of the lowest term. In these cases the user should specify the multiplicity corresponding to a chosen value of pseudospin ($2\tilde{S}+1$). For instance, in Dy^{3+} the spin of the ground state term is $S = 5/2$, but in most of real compounds only the ground Kramers doublet is considered. In such case, the multiplicity of the pseudospin equals to 2 (see `MLTP` keyword). For the calculation of the parameters of the crystal field corresponding to the ground atomic `multiplet` J for lanthanides should be requested with the keywords `CRYS_element` and `CRYS_charge`. Similarly, the parameters of the crystal field corresponding to the ground atomic `term` L for lanthanides and transition metals compounds can be requested with same keywords: `CRYS_element` and `CRYS_charge`.

Note that the keywords/syntax in the ORCA CASSCF/ANISO block are slightly different from the genuine `SINGLE_ANISO` input, where some of the keywords are grouped together. We aimed at keeping the control keywords as close as possible.

Optional general keywords to control the input within the ORCA interface (CASSCF/ANISO subblock):

`InputNameOnDisk`

This keyword reads the name of the input file for `SINGLE_ANISO`, a string given between quotations. Example:

```
InputNameOnDisk "my_input_for_aniso.inp"
```

The interface with ORCA will add the `DATA` keyword with specific name of the datafile for the performed calculation. All the other keywords provided inside this file must follow their original names, as in `MOLCAS`.

`MLTP`

The number of molecular multiplets ($i.e.$ groups of spin-orbital eigenstates) for which g, D and higher magnetic tensors will be calculated (default `MLTP`=1). With `MLTP` an comma separated list of numbers specifying the dimension of each multiplet is passed. The default is to select one multiplet which has the dimension equal

to the multiplicity of the ground term. In cases of strong spin-orbit coupling the usage of this keyword is mandatory. Example:

```
MLTP 4, 4, 2, 2
```

SINGLE_ANISO will compute the g tensor for 4 groups of states: 2 groups having the effective spin $S = |3/2>$, and other 2 groups of states being Kramers doublets.

TINT

Specifies the temperature points for the evaluation of the magnetic susceptibility. The program will read three numbers: T_{min}, T_{max}, and nT.

- T_{min} - the minimal temperature (Default 0.0K)
- T_{max} - the maximal temperature (Default 300.0K)
- nT - number of temperature points (Default 101)

Example:

```
TINT 0.0, 330.0, 331
```

SINGLE_ANISO will compute temperature dependence of the magnetic susceptibility in 331 points evenly distributed in temperature interval: 0.0K - 330.0K.

HINT

Specifies the field points for the evaluation of the magnetisation in a certain direction. The program will read three numbers: H_{min}, H_{max} and nH.

- H_{min} - the minimal field (Default 0.0T)
- H_{max} - the maximal filed (Default 10.0T)
- nH - number of field points (Default 101)

Example:

```
HINT 0.0, 20.0, 201
```

SINGLE_ANISO will compute the molar magnetisation in 201 points evenly distributed in field interval: 0.0T - 20.0T.

TMAG

Specifies the temperature(s) at which the field-dependent magnetisation is calculated. The program will read the temperatures (in Kelvin) at which magnetisation is to be computed. Default is to compute magnetisation at one temperature point (2.0 K). Example:

```
TMAG 1.8, 2.0, 3.0, 4.0, 5.0
```

SINGLE_ANISO will compute the molar magnetisation at 5 temperature points (1.8 K, 2.0 K, 3.4 K, 4.0 K, and 5.0 K).

ENCU

The keyword expects to read two integer numbers. The two parameters (NK and MG) are used to define the cut-off energy for the lowest states for which Zeeman interaction is taken into account exactly. The contribution to the magnetisation coming from states that are higher in energy than E (see below) is done by second order perturbation theory. The program will read two integer numbers: NK and MG. Default values are: $NK = 100, MG = 100$.

$$E = NK \cdot k_{Boltz} \cdot \mathrm{TMAG}_{max} + MG \cdot \mu_{Bohr} \cdot H_{max}$$

The field-dependent magnetisation is calculated at the maximal temperature value given by TMAG keyword. Example:

```
ENCU 250, 150
```

If H_{max} = 10 T and TMAG = 1.8 K, then the cut-off energy is: $$E = 250 \cdot k_{Boltz} \cdot 1.8 + 150 \cdot \mu_{Bohr} \cdot 10 = 1013.06258(cm^{-1})$$

This means that the magnetisation arising from all spin-orbit states with energy lower than $E = 1013.06258(cm^{-1})$ will be computed exactly (i.e. are included in the exact Zeeman diagonalisation) The keywords NCUT, ERAT and ENCU have similar purpose. If two of them are used at the same time, the following priority is defined: NCUT > ENCU > ERAT.

NCUT

This flag is used to define the cut-off energy for the low-lying spin-orbit states for which Zeeman interaction is taken into account exactly. The contribution to the magnetisation arising from states that are higher in energy than lowest N_{CUT} states, is done by second-order perturbation theory. The program will read one integer number. In case the number is larger than the total number of spin-orbit states(N_{SS}, then the N_{CUT} is set to N_{SS} (which means that the molar magnetisation will be computed exactly, using full Zeeman diagonalisation for all field points). The field-dependent magnetisation is calculated at the temperature value(s) defined by TMAG. Example:

```
NCUT 32
```

The keywords NCUT, ERAT and ENCU have similar purpose. If two of them are used at the same time, the following priority is defined: NCUT > ENCU > ERAT.

ERAT

This flag is used to define the cut-off energy for the low-lying spin-orbit states for which Zeeman interaction is taken into account exactly. The program will read one single real number specifying the ratio of the energy states which are included in the exact Zeeman Hamiltonian. As example, a value of 0.5 means that the lowest half of the energy states included in the spin-orbit calculation are used for exact Zeeman diagonalisation.

Example:

```
ERAT 0.333
```

The keywords NCUT, ERAT and ENCU have similar purpose. If two of them are used at the same time, the following priority is defined: NCUT > ENCU > ERAT.

MVEC_x MVEC_y MVEC_z

MVEC_x, MVEC_y and MVEC_z define a number of directions for which the magnetisation vector will be computed. The directions are given as unitary vectors specifying the direction *i* of the applied magnetic field).

Example:

```
MVEC_x 0.00, 1.57, 1.57, 0.425
MVEC_y 0.00, 0.00, 1.57, 0.418
MVEC_z 0.00, 0.00, 1.57, 0.418
```

ZEEM

This keyword allows to compute Zeeman splitting spectra along certain directions of applied field. Directions of applied field are given as three real number for each direction, specifying the projections along each direction: Example:

```
ZEEM[0] 1.0, 0.0, 0.0
ZEEM[1] 0.0, 1.0, 0.0
ZEEM[2] 0.0, 0.0, 1.0
ZEEM[3] 0.0, 1.0, 1.0
ZEEM[4] 1.0, 0.0, 1.0
ZEEM[5] 1.0, 1.0, 0.0
```

The above input will request computation of the Zeeman spectra along six directions: Cartesian axes X, Y, Z (directions 1,2 and 3), and between any two Cartesian axes: YZ, XZ and XY, respectively. The program will renormalise the input vectors according to unity length. In combination with PLOT keyword, the corresponding zeeman_energy_xxx.png images will be produced.

MAVE

The keyword requires two integer numbers, denoted MAVE_nsym and MAVE_ngrid. The parameters MAVE_nsym and MAVE_ngrid specify the grid density in the computation of powder molar magnetisation. The program uses Lebedev-Laikov distribution of points on the unit sphere. The parameters are integer numbers: n_{sym} and n_{grid}. The n_{sym} defines which part of the sphere is used for averaging. It takes one of the three values: 1 (half-sphere), 2 (a quarter of a sphere) or 3 (an octant of the sphere). n_{grid} takes values from 1 (the smallest grid) till 32 (the largest grid, i.e. the densest). The default is to consider integration over a half-sphere (since $M(H) = -M(-H)$): $n_{sym} = 1$ and $n_{sym} = 15$ (i.e 185 points distributed over half-sphere). In case of symmetric compounds, powder magnetisation may be averaged over a smaller part of the sphere, reducing thus the number of points for the integration. The user is responsible to choose the appropriate integration scheme. Note that the program's default is rather conservative.

Example:

```
MAVE 1, 8
```

TEXP_temp TEXP_chiT

The parameters TEXP_temp and TEXP_chiT allow the computation of the magnetic susceptibility $\chi T(T)$ at experimental points. The experimental temperature (in K) and the experimental magnetic susceptibility (in $cm^3 K mol^{-1}$) are read as comma separated list. In the case both TEXP and TINT keywords are given, the TEXP will be used while the TINT input will be ignored.

Example:

```
TEXP_temp 0.0, 10.0, 20.0, 30.0, 40.0, 50.0
TEXP_chiT 4.5, 4.5, 4.58, 4.62, 4.66, 4.70
```

HEXP_temp HEXP_H HEXP_M

The three keywords HEXP_temp, HEXP_H and HEXP_M enable the computation of the molar magnetisation $M_{mol}(H)$ at experimental points. The experimental field strength (in Tesla) and the experimental magnetisation (in μ_{Bohr}) are read as a comma separated list. In the case both HEXP and HINT keywords are given, the HEXP will be used while the HINT input will be ignored. The magnetisation routine will print the standard deviation from the experiment. Example:

```
HEXP_temp 2.0, 3.0
HEXP_H 0.0, 1.0, 2.0, 3.0, 4.0
HEXP_M[0]= 0.0, 2.46, 2.86, 2.95, 2.98  # exp. M at T=2.0 K
HEXP_M[1]= 0.0, 2.04, 2.68, 2.87, 2.94  # exp. M at T=3.0 K
```

ZJPR

This keyword specifies the value (in cm^{-1}) of a phenomenological parameter of a mean molecular field acting on the spin of the complex (the average intermolecular exchange constant). It is used in the calculation of all magnetic properties (not for spin Hamiltonians) (Default is 0.0).

```
ZJPR -0.02
```

TORQ

This keyword specifies the number of angular points for the computation of the magnetisation torque function, $\vec{\tau}_\alpha$ as function of the temperature, field strength and field orientation.

```
TORQ 55
```

The torque is computed at all temperature given by TMAG or HEXP_temp inputs. Three rotations around Cartesian axes X, Y and Z are performed.

PrintLevel

This keyword controls the print level.

- 2 - normal. (Default)
- 3 or larger (debug)

CRYS_element CRYS_charge

The keywords `CRYS_element` and `CRYS_charge` request the computation of all 27 Crystal-Field parameters acting on the ground atomic multiplet of a lanthanide. With `CRYS_element` the chemical symbol of the lanthanide is set. **Note that the element symbol must be enclosed in quotation marks**. The charge is defined with `CRYS_charge`. By default the program will not compute the parameters of the Crystal-Field.

Example:

```
CRYS_element "Dy" CRYS_charge 3
```

QUAX

This keyword controls the quantisation axis for the computation of the Crystal-Field parameters acting on the ground atomic multiplet of a lanthanide. On the next line, the program will read one of the three values: 1, 2 or 3.

- 1 - quantisation axis is the main magnetic axis Zm of the ground pseudospin multiplet, whose size is specified within the `MLTP` keyword. (Default)
- 2 - quantisation axis is the main magnetic axis Zm of the entire atomic multiplet $|J, M_J >$.
- 3 - quantisation axis is the original Cartesian Z axis. Rotation matrix is unity.

Example:

```
QUAX 3
```

UBAR

With `UBAR` set to "true", the blocking barrier of a single-molecule magnet is estimated. The default is not to compute it. The method prints transition matrix elements of the magnetic moment according to the Figure below:

In this figure, a qualitative performance picture of the investigated single-molecular magnet is estimated by the strengths of the transition matrix elements of the magnetic moment connecting states with opposite magnetisaskytions ($n+ \rightarrow n-$). The height of the barrier is qualitatively estimated by the energy at which the

matrix element ($n+ \rightarrow n-$) is large enough to induce significant tunnelling splitting at usual magnetic fields (internal) present in the magnetic crystals (0.01-0.1 Tesla). For the above example, the blocking barrier closes at the state ($8+ \rightarrow 8-$).

All transition matrix elements of the magnetic moment are given as $((|\mu_X| + |\mu_Y| + |\mu_Z|)/3)$. The data is given in Bohr magnetons (μ_{Bohr}).

Example:

```
UBAR true
```

ABCC_abc ABCC_center

The keywords `ABCC_abc` and `ABCC_center` set the computation of magnetic and anisotropy axes in the crystallographic abc system. With `ABCC_abc`, the program reads six real values, namely a, b, c, α, β, and γ, defining the crystal lattice. The values must be separated by a comma. With `ABCC_center`, the program reads the fractional coordinates of the magnetic center (from the CIF file) - again separated by comma. It is assumed that the XYZ coordinates used for the *ab initio* calculations did not rotate or translate the molecule from its crystallographic position. This input will ensure that all tensors computed by `SINGLE_ANISO` are given also in the abc system. The computed values in the output correspond to the crystallographic position of three "dummy atoms" located on the corresponding anisotropy axes, at the distance of 1.0 $\AA$ from the metal site. Example:

```
ABCC_abc 12.977, 12.977, 16.573, 90, 90, 120
ABCC_center 0.666667, 0.333333, 0.20413
```

XFIE

This keyword specifies the value (in T) of applied magnetic field for the computation of magnetic susceptibility by dM/dH and M/H formulas. A comparison with the usual formula (in the limit of zero applied field) is provided. (Default is 0.0). Example:

```
XFIE 0.35
```

This keyword together with the keyword `PLOT` will enable the generation of two additional plots: `XT_with_field_dM_over_dH.png` and `XT_with_field_M_over_H.png`, one for each of the two above formula used, alongside with respective `gnuplot` scripts and gnuplot datafiles.

PLOT

Set to "true", the program generates a few plots (png or eps format) via an interface to the linux program *gnuplot*. The interface generates a datafile, a *gnuplot* script and attempts execution of the script for generation of the image. The plots are generated only if the respective function is invoked. The magnetic susceptibility, molar magnetisation and blocking barrier (`UBAR`) plots are generated. The files are named: `XT_no_field.dat`, `XT_no_field.plt`, `XT_no_field.png`, `MH.dat`, `MH.plt`, `MH.png`, `BARRIER_TME.dat`, `BARRIER_ENE.dat`, `BARRIER.plt` and `BARRIER.png`, `zeeman_energy_xxx.png` etc. All files produced by `SINGLE_ANISO` are referenced in the corresponding output section. Example:

```
PLOT true
```

5.34.4 How to cite

We would appreciate if you cite the following papers in publications resulting from the use of `SINGLE_ANISO`:

- Chibotaru, L. F.; Ungur, L. *J. Chem. Phys.*, **2012**, *137*, 064112.
- Ungur, L. Chibotaru, L. F. *Chem. Eur. J.*, **2017**, *23*, 3708-3718.

In addition, useful information like the definition of pseudospin Hamiltonians and their derivation can be found in this paper.

5.35 Interface to POLY_ANISO Module

5.35.1 General Description

The POLY_ANISO is a stand-alone utility allowing for a semi-*ab initio* description of the (low-lying) electronic structure and magnetic properties of polynuclear compounds. The model behind it is based on the *localised* nature of the magnetic orbitals (i.e. the d and f orbitals containing unpaired electrons). For many compounds of interest, the localised character of the magnetic orbitals leads to very weak character of the exchange interaction between magnetic centers. Due to this weakness of the inter-site interaction, the molecular orbitals and corresponding localised ground and excited states may be optimized in the absence of the magnetic interaction at all. For this purpose, various fragmentation models may be applied. The most commonly used fragmentation model is exemplified in Fig. 5.66:

Fig. 5.66: Fragmentation model of a polynuclear compound.

The upper scheme shows a schematic overview of a tri-nuclear compound and the resulting three mononuclear fragments obtained by *diamagnetic atom substitution method*. By this scheme, the neighbouring magnetic centers, containing unpaired electrons are computationally replaced by their diamagnetic equivalents. As example, transition metal (TM) sites are best described by either a diamagnetic Zn(II) or Sc(III), in function of which one is the closest (in terms of charge and atomic radius). For lanthanides (LN), the same principle is applicable, La(III), Lu(III) or Y(III) are best suited to replace a given magnetic lanthanide. Individual mononuclear metal fragments are then investigated by the common CASSCF+SOC/NEVPT2+ SOC/SINGLE_ANISO computational method. A single datafile for each magnetic site, produced by the SINGLE_ANISO run, is needed by the POLY_ANISO code as input.

Magnetic interaction between metal sites is very important for accurate description of low-lying states and their properties. While the full exchange interaction is quite complex (e.g. requiring a multipolar description employing a large set of parameters [755, 756]), in a simplified model it can be viewed as a sum of various interaction mechanisms: magnetic exchange, dipole-dipole interaction, antisymmetric exchange, etc. In the POLY_ANISO code we have implemented several mechanisms, which can be invoked simultaneously for each interacting pair.

The description of the magnetic exchange interaction is done within the Lines model[757]. This model is exact in three cases:

1. interaction between two isotropic spins (Heisenberg),
2. interaction between one Ising spin (only S_Z component) and one isotropic (i.e. usual) spin, and
3. interaction between two Ising spins.

In all other cases when magnetic sites have intermediate anisotropy (i.e. when the spin-orbit coupling and crystal field effects are of comparable strengths), the Lines model represents an approximation. However, it was successfully applied for a wide variety of polynuclear compounds so far.

In addition to the magnetic exchange, magnetic dipole-dipole interaction can be accounted exactly, by using the *ab initio* computed magnetic moment for each metal site (as available inside the datafile). In the case of strongly anisotropic lanthanide compounds (like Ho^{3+} or Dy^{3+}), the magnetic dipole-dipole interaction is usually the dominant one. For example, a system containing two magnetic dipoles $\vec{\mu}_1$ and $\vec{\mu}_2$, separated by distance $\vec{r}$ have a total

energy:

$$E_{dip} = \frac{\mu_{Bohr}^2}{r^3}[\vec{\mu}_1 \cdot \vec{\mu}_2 - 3(\vec{\mu}_1\vec{n}_{1,2}) \cdot (\vec{\mu}_2\vec{n}_{1,2})],$$

where $\vec{\mu}_{1,2}$ are the magnetic moments of sites 1 and 2, respectively; r is the distance between the two magnetic dipoles, $\vec{n}_{1,2}$ is the directional vector connecting the two magnetic dipoles (of unit length). μ_{Bohr}^2 is the square of the Bohr magneton; with an approximate value of 0.43297 in cm^{-1}/T. As inferred from the above Equation, the dipolar magnetic interaction depends on the distance and on the angle between the magnetic moments on magnetic centers. Therefore, the Cartesian coordinates of all non-equivalent magnetic centers must be provided in the input.

In brief, the `POLY_ANISO` is performing the following operations:

1. read the input and information from the datafiles
2. build the exchange coupled basis
3. compute the magnetic exchange, magnetic dipole-dipole, and other magnetic Hamiltonians using the *ab initio*-computed spin and orbital momenta of individual magnetic sites and the input parameters
4. sum up all the magnetic interaction Hamiltonians and diagonalise the total interaction Hamiltonian
5. rewrite the spin and magnetic moment in the exchange-coupled eigenstates basis
6. use the obtained spin and magnetic momenta for the computation of the magnetic properties of entire polynuclear compound

The actual values of the inter-site magnetic exchange could be derived from e.g. broken-symmetry DFT calculations. Alternatively, they could be regarded as fitting parameters, while their approximate values could be extracted by minimising the standard deviation between measured and calculated magnetic data.

5.35.2 Files

`POLY_ANISO` is called independently of ORCA for now. In the future versions of ORCA we will aim for a deeper integration, for a better experience.

```
bash:$
bash:$  $ORCA/x86_64/otool_poly_aniso  <  poly_aniso.input   >  poly_aniso.output
bash:$
```

The actual names of the `poly_aniso.input` and `poly_aniso.output` are not hard coded, and can take any names. A bash script for a more convenient usage of `POLY_ANISO` can be provided upon request or made available on the Forum.

Input Files

The program `POLY_ANISO` needs the following files:

`aniso_i.input`
This is an ASCII text file generated by the `CASSCF/SOC/ SINGLE_ANISO` run. It should be provided for `POLY_ANISO` as `aniso_i.input` (i=1,2,3, etc.): one file for each magnetic center. In cases when the entire polynuclear cluster or molecule has exact point group symmetry, only `aniso_i.input` files for crystallographically non-equivalent centers should be given. This saves computational time since equivalent metal sites do not need to be computed *ab initio*.

`poly_aniso.input`
The standard input file defining the computed system and various input parameters. This file can take any name.

Output Files

5.35.3 List of Keywords

This section describes the keywords used to control the POLY_ANISO input file. Only two keywords NNEQ, PAIR (and SYMM if the polynuclear cluster has symmetry) are mandatory for a minimal execution of the program, while the other keywords allow customisation of the execution of the POLY_ANISO.

The format of the "poly_aniso.input" file resembles to a certain extent the input file for SINGLE_ANISO program. The input file must start with "&POLY_ANISO" text.

Mandatory Keywords Defining the Calculation

Keywords defining the polynuclear cluster:

NNEQ This keyword defines several important parameters of the calculation. On the first line after the keyword the program reads 2 values: 1) the number of types of different magnetic centers (NON-EQ) of the cluster and 2) a letter T or F in the second position of the same line. The number of NON-EQ is the total number of magnetic centers of the cluster which cannot be related by point group symmetry. In the second position the answer to the question: "Have all NON-EQ centers been computed *ab initio*?" is given: T for True and F for False. On the following line the program will read NON-EQ values specifying the number of equivalent centers of each type. On the following line the program will read NON-EQ integer numbers specifying the number of low-lying spin-orbit functions from each center forming the local exchange basis.

Some examples valid for situations where all sites have been computed *ab initio* (case T, True):

```
NNEQ
  2  T
  1  2
  2  2
```

There are two kinds of magnetic centers in the cluster; both have been computed *ab initio*; the cluster consists of 3 magnetic centers: one center of the first kind and two centers of the second kind. From each center we take into the exchange coupling only the ground doublet. As a result, the $N_{exch} = 2^1 \times 2^2 = 8$, and the two datafiles aniso_1.input (for-type 1) and aniso_2.input (for-type 2) files must be present.

```
NNEQ
  3  T
  2  1  1
  4  2  3
```

There are three kinds of magnetic centers in the cluster; all three have been computed *ab initio*; the cluster consists of four magnetic centers: two centers of the first kind, one center of the second kind and one center of the third kind. From each of the centers of the first kind we take into exchange coupling four spin-orbit states, two states from the second kind and three states from the third center. As a result the $N_{exch} = 4^2 \times 2^1 \times 3^1 = 96$. Three files aniso_i.input for each center ($i = 1, 2, 3$) must be present.

```
NNEQ
  6  T
  1  1  1  1  1  1
  2  4  3  5  2  2
```

There are six kinds of magnetic centers in the cluster; all six have been computed *ab initio*; the cluster consists of 6 magnetic centers: one center of each kind. From the center of the first kind we take into exchange coupling two spin-orbit states, four states from the second center, three states from the third center, five states from the fourth center and two states from the fifth and sixth centers. As a result the $N_{exch} = 2^1 \times 4^1 \times 3^1 \times 5^1 \times 2^1 \times 2^1 = 480$. Six files aniso_i.input for each center ($i = 1, 2, ..., 6$) must be present.

Only in cases when some centers have NOT been computed *ab initio* (i.e. for which no aniso_i.input file exists), the program will read an additional line consisting of NON-EQ letters (A or B) specifying the type of each of the NON-EQ centers: A - the center is computed *ab initio* and B - the center is considered isotropic. On the

following number-of-B-centers line(s) the isotropic g factors of the center(s) defined as B are read. The spin of the B center(s) is defined: $S = (N-1)/2$, where N is the corresponding number of states to be taken into the exchange coupling for this particular center. Some examples valid for mixed situations: the system consists of centers computed *ab initio* and isotropic centers (case F, False):

```
NNEQ
  2  F
  1  2
  2  2
  A  B
  2.3   2.3   2.3
```

There are two kinds of magnetic centers in the cluster; the center of the first type has been computed *ab initio*, while the centers of the second type are considered isotropic with $g_X = g_Y = g_Z$=2.3; the cluster consists of three magnetic centers: one center of the first kind and two centers of the second kind. Only the ground doublet state from each center is considered for the exchange coupling. As a result the $N_{exch} = 2^1 \times 2^2 = 8$. File `aniso_i.input` (for-type 1) must be present.

```
NNEQ
  3  F
  2  1  1
  4  2  3
  A  B  B
  2.3  2.3  2.0
  2.0  2.0  2.5
```

There are three kinds of magnetic centers in the cluster; the first center type has been computed *ab initio*, while the centers of the second and third types are considered empirically with $g_X = g_Y$ =2.3; g_Z=2.0 (second type) and $g_X = g_Y$ =2.0; g_Z=2.5 (third type); the cluster consists of four magnetic centers: two centers of the first kind, one center of the second kind and one center of the third kind. From each of the centers of the first kind, four spin-orbit states are considered for the exchange coupling, two states from the second kind and three states from the center of the third kind. As a result the $N_{exch} = 4^2 \times 2^1 \times 3^1 = 96$. The file `aniso_i.input` must be present.

```
NNEQ
  6  T
  1  1  1  1  1  1
  2  4  3  5  2  2
  B  B  A  A  B  A
  2.12  2.12  2.12
  2.43  2.43  2.43
  2.00  2.00  2.00
```

There are six kinds of magnetic centers in the cluster; only three centers have been computed *ab initio*, while the other three centers are considered isotropic; the g factors of the first center is 2.12 (S=1/2); of the second center 2.43 (S=3/2); of the fifth center 2.00 (S=1/2); the entire cluster consists of six magnetic centers: one center of each kind. From the center of the first kind, two spin-orbit states are considered in the exchange coupling, four states from the second center, three states from the third center, five states from the fourth center and two states from the fifth and sixth centers. As a result the $N_{exch} = 2^1 \times 4^1 \times 3^1 \times 5^1 \times 2^1 \times 2^1 = 480$. Three files `aniso_3.input` and `aniso_4.input` and `aniso_6.input` must be present.

There is no maximal value for `NNEQ`, although the calculation becomes quite heavy in case the number of exchange functions is large.

`SYMM` Specifies rotation matrices to symmetry equivalent sites. This keyword is mandatory in the case more centers of a given type are present in the calculation. This keyword is mandatory when the calculated polynuclear compound has exact crystallographic point group symmetry. In other words, when the number of equivalent centers of any kind i is larger than 1, this keyword must be employed. Here the rotation matrices from the one center to all the other of the same type are declared. On the following line the program will read the number 1 followed on the next lines by as many 3×3 rotation matrices as the total number of equivalent centers of type 1. Then the rotation matrices of centers of type 2, 3 and so on, follow in the same format. When the rotation matrices contain irrational numbers (e.g. $\sin \frac{\pi}{6} = \frac{\sqrt{3}}{2}$), then more digits than presented in the examples below are advised to be given: $\frac{\sqrt{3}}{2} = 0.8660254$.

Examples:

```
NNEQ
  2  F
  1  2
  2  2
  A  B
  2.3  2.3  2.3
  SYMM
  1
  1.0 0.0 0.0
  0.0 1.0 0.0
  0.0 0.0 1.0
  2
  1.0 0.0 0.0
  0.0 1.0 0.0
  0.0 0.0 1.0
 -1.0 0.0 0.0
  0.0 -1.0 0.0
  0.0 0.0 -1.0
```

The cluster computed here is a tri-nuclear compound, with one center computed *ab initio*, while the other two centers, related to each other by inversion, are considered isotropic with $g_X = g_Y = g_Z = 2.3$. The rotation matrix for the first center is I (identity, unity) since the center is unique. For the centers of type 2, there are two matrices 3×3 since we have two centers in the cluster. The rotation matrix of the first center of type 2 is Identity while the rotation matrix for the equivalent center of type 2 is the inversion matrix.

```
NNEQ
  3  F
  2  1  1
  4  2  3
  A  B  B
  2.1 2.1 2.1
  2.0 2.0 2.0
  SYMM
  1
  1.0 0.0 0.0
  0.0 1.0 0.0
  0.0 0.0 1.0
  0.0 -1.0 0.0
  -1.0 0.0 0.0
  0.0 0.0 1.0
  2
  1.0 0.0 0.0
  0.0 1.0 0.0
  0.0 0.0 1.0
  3
  1.0 0.0 0.0
  0.0 1.0 0.0
  0.0 0.0 1.0
```

In this input a tetranuclear compound is defined, all centers are computed *ab initio*. There are two centers of type "1", related one to each other by C_2 symmetry around the Cartesian Z axis. Therefore the `SYMM` keyword is mandatory. There are two matrices for centers of type 1, and one matrix (identity) for the centers of type 2 and type 3.

```
NNEQ
  6  F
  1  1  1  1  1  1
  2  4  3  5  2  2
  B  B  A  A  B  A
  2.12 2.12 2.12
  2.43 2.43 2.43
```

(continues on next page)

(continued from previous page)

```
  2.00 2.00 2.00
```

In this case the computed system has no symmetry. Therefore, the `SYMM` keyword is not required. `End of Input` Specifies the end of the input file. No keywords after this one will be processed.

Keywords Defining the Magnetic Exchange Interactions

This section defines the keywords used to set up the interacting pairs of magnetic centers and the corresponding exchange interactions.

A few words about the numbering of the magnetic centers of the cluster in the `POLY_ANISO`. First all equivalent centers of the type 1 are numbered, then all equivalent centers of the type 2, etc. These labels of the magnetic centers are used further for the declaration of the magnetic coupling.

PAIR or LIN1

This keyword defines the interacting pairs of magnetic centers and the corresponding exchange interaction. A few words about the numbering of the magnetic centers of the cluster in the `POLY_ANISO`. First all equivalent centers of the type 1 are numbered, then all equivalent centers of the type 2, etc. These labels of the magnetic centers are used now for the declaration of the magnetic coupling. Interaction Hamiltonian is:

$$\hat{H}_{Lines} = -\sum_{p=1}^{N_{pairs}} J_p \hat{s}_i \hat{s}_j,$$

where i an j are the indices of the metal sites of the interacting pair p; J_p is the user-defined magnetic exchange interaction between the corresponding metal sites; $\hat{s}_i$ and $\hat{s}_j$ are the `ab initio` spin operators for the low-lying exchange eigenstates.

```
PAIR
3
1 2 -0.2
1 3 -0.2
2 3  0.4
```

The input above is applicable for a tri-nuclear molecule. Two interactions are antiferromagnetic while ferromagnetic interaction is given for the last interacting pair.

LIN3

This keyword defines a more involved exchange interaction, where the user is allowed to define 3 parameters for each interacting pair. The interaction Hamiltonian is given by:

$$\hat{H}_{Lines} = -\sum_{p=1}^{N_{pairs}} \sum_{\alpha} J_{p,\alpha} \hat{s}_{i,\alpha} \hat{s}_{j,\alpha},$$

where the α defines the Cartesian axis x, y, z.

```
LIN3
1
1 2 -0.2 -0.4 -0.6 # i, j, Jx, Jy, Jz
```

The input above is applicable for a mononuclear molecule.

LIN9

This keyword defines a more involved exchange interaction, where the user is allowed to define 9 parameters for each interacting pair. The interaction Hamiltonian is given by:

$$\hat{H}_{Lines} = -\sum_{p=1}^{N_{pairs}} \sum_{\alpha,\beta} J_{p,\alpha,\beta} \hat{s}_{i,\alpha} \hat{s}_{j,\beta},$$

where the α and β defines the Cartesian axis x, y, z.

```
LIN9
1
1 2 -0.1 -0.2 -0.3 -0.4 -0.5 -0.6 -0.7 -0.8 -0.9
# i,j,Jxx,Jxy,Jxz,Jyx,Jyy,Jyz,Jzx,Jzy,Jzz
```

The input above is applicable for a mononuclear molecule.

COOR

The COOR keyword turns ON the computation of the dipolar coupling for those interacting pairs which were declared under PAIR, LIN3 or LIN9 keywords. On the NON-EQ lines following the keyword the program will read the symmetrised Cartesian coordinates of NON-EQ magnetic centers: one set of symmetrised Cartesian coordinates for each type of magnetic centers of the system. The *symmetrized Cartesian coordinates* are obtained by translating the original coordinates to the origin of Coordinate system, such that by applying the corresponding SYMM rotation matrix onto the input COOR data, the position of all other sites are generated.

```
COOR
6.489149    3.745763    1.669546
5.372478    5.225861    0.505625
```

The magnetic dipole-dipole Hamiltonian is computed as follows:

$$\hat{H}_{dip} = \mu_{Bohr}^2 \sum_{p=1}^{N_{pairs}} \frac{\hat{\mu}_i\hat{\mu}_j - 3(\hat{\mu}_i\vec{n}_{i,j})(\hat{\mu}_j\vec{n}_{i,j})}{\vec{r_{i,j}^3}}$$

and is added to $\hat{H}_{exch}$ computed using other models. The $\hat{H}_{dip}$ is added for all magnetic pairs.

Optional General Keywords to Control the Input

Normally POLY_ANISO runs without specifying any of the following keywords. However, some properties are only computed if it is requested by the respective keyword. Argument(s) to the keyword are always supplied on the next line of the input file.

MLTP

The number of molecular multiplets (i.e. groups of spin-orbital eigenstates) for which g, D and higher magnetic tensors will be calculated (default MLTP=1). The program reads two lines: the first is the number of multiplets (N_{MULT}) and the second the array of N_{MULT} numbers specifying the dimension (multiplicity) of each multiplet. Example:

```
MLTP
10
2 4 4 2 2 2 2 2 2 2
```

POLY_ANISO will compute the EPR g and D- tensors for 10 groups of states. The groups 1 and 4-10 are doublets ($\tilde{S} = |1/2\rangle$), while the groups 2 and 3 are quadruplets, having the effective spin $\tilde{S} = |3/2\rangle$. For the latter cases, the ZFS (D-) tensors will be computed. We note here that large degeneracies are quite common for exchange coupled systems, and the data for this keyword can only be rendered after the inspection of the exchange spectra.

TINT

Specifies the temperature points for the evaluation of the magnetic susceptibility. The program will read three numbers: T_{min}, T_{max}, and nT.

- T_{min} - the minimal temperature (Default 0.0 K)
- T_{max} - the maximal temperature (Default 300.0 K)
- nT - number of temperature points (Default 301)

Example:

```
TINT
0.0 300.0 331
```

POLY_ANISO will compute temperature dependence of the magnetic susceptibility in 331 points evenly distributed in temperature interval: 0.0 K - 330.0 K.

HINT

Specifies the field points for the evaluation of the molar magnetisation. The program will read three numbers: H_{min}, H_{max}, nH.

- H_{min} - the minimal field (Default 0.0 T)
- H_{max} - the maximal filed (Default 10.0 T)
- nH - number of field points (Default 101)

Example:

```
HINT
0.0 20.0 201
```

POLY_ANISO will compute the molar magnetisation in 201 points evenly distributed in field interval: 0.0 T - 20.0 T.

TMAG

Specifies the temperature(s) at which the field-dependent magnetisation is calculated. Default is one temperature point, T = 2.0 K.

Example:

```
TMAG
6   1.8 2.0 2.4 2.8 3.2 4.5
```

ENCU

The keyword expects to read two integer numbers. The two parameters (NK and MG) are used to define the cut-off energy for the lowest states for which Zeeman interaction is taken into account exactly. The contribution to the magnetisation coming from states that are higher in energy than E (see below) is done by second order perturbation theory. The program will read two integer numbers: NK and MG. Default values are: $NK = 100, MG = 100$.

$$E = NK \cdot k_{Boltz} \cdot \mathtt{TMAG}_{max} + MG \cdot \mu_{Bohr} \cdot H_{max}$$

The field-dependent magnetisation is calculated at the maximal temperature value given by TMAG keyword. Example:

```
ENCU
250 150
```

If H_{max} = 10 T and TMAG = 1.8 K, then the cut-off energy is:

$$E = 250 \cdot k_{Boltz} \cdot 1.8 + 150 \cdot \mu_{Bohr} \cdot 10 = 1013.06258(cm^{-1})$$

This means that the magnetisation arising from all exchange states with energy lower than $E = 1013.06258(cm^{-1})$ will be computed exactly (i.e. are included in the exact Zeeman diagonalisation) The keywords NCUT, ERAT and ENCU have similar purpose. If two of them are used at the same time, the following priority is defined: NCUT > ENCU > ERAT.

UBAR

With UBAR set to "true", the blocking barrier of a single-molecule magnet is estimated. The default is not to compute it. The method prints transition matrix elements of the magnetic moment according to the Figure below:

In this figure, a qualitative performance picture of the investigated single-molecular magnet is estimated by the strengths of the transition matrix elements of the magnetic moment connecting states with opposite magnetisations ($n+ \rightarrow n-$). The height of the barrier is qualitatively estimated by the energy at which the matrix element ($n+ \rightarrow n-$) is large enough to induce significant tunnelling splitting at usual magnetic fields (internal) present in the magnetic crystals (0.01-0.1 Tesla). For the above example, the blocking barrier closes at the state ($8+ \rightarrow 8-$). All transition matrix elements of the magnetic moment are given as $((|\mu_X| + |\mu_Y| + |\mu_Z|)/3)$. The data is given in Bohr magnetons (μ_{Bohr}). Example:

```
UBAR
```

ERAT

This flag is used to define the cut-off energy for the low-lying exchange-coupled states for which Zeeman interaction is taken into account exactly. The program will read one single real number specifying the ratio of the energy states which are included in the exact Zeeman Hamiltonian. As example, a value of 0.5 means that the lowest half of the energy states included in the spin-orbit calculation are used for exact Zeeman diagonalisation. Example:

```
ERAT
0.333
```

The keywords `NCUT`, `ERAT` and `ENCU` have similar purpose. If two of them are used at the same time, the following priority is defined: `NCUT > ENCU > ERAT`.

NCUT

This flag is used to define the cut-off energy for the low-lying exchange states for which Zeeman interaction is taken into account exactly. The contribution to the magnetisation arising from states that are higher in energy than lowest N_{CUT} states, is done by second-order perturbation theory. The program will read one integer number. In case the number is larger than the total number of exchange states(N_{exch}, then the N_{CUT} is set to N_{SS} (which means that the molar magnetisation will be computed exactly, using full Zeeman diagonalisation for all field points). The field-dependent magnetisation is calculated at the temperature value(s) defined by `TMAG`. Example:

```
NCUT
32
```

The keywords NCUT, ERAT and ENCU have similar purpose. If two of them are used at the same time, the following priority is defined: NCUT > ENCU > ERAT.

MVEC

MVEC, define a number of directions for which the magnetisation vector will be computed. The directions are given as vectors specifying the direction *i* of the applied magnetic field).

Example:

```
MVEC
4 # number of directions
1.0 0.0 0.0  # px, py, pz of each direction
0.0 1.0 0.0
0.0 0.0 1.0
1.0 1.0 1.0
```

ZEEM

This keyword allows to compute Zeeman splitting spectra along certain directions of applied field. Directions of applied field are given as three real number for each direction, specifying the projections along each direction: Example:

```
ZEEM
6
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
0.0 1.0 1.0
1.0 0.0 1.0
1.0 1.0 0.0
```

The above input will request computation of the Zeeman spectra along six directions: Cartesian axes X, Y, Z (directions 1,2 and 3), and between any two Cartesian axes: YZ, XZ and XY, respectively. The program will renormalise the input vectors according to unity length. In combination with PLOT keyword, the corresponding zeeman_energy_xxx.png images will be produced.

MAVE

The keyword requires two integer numbers, denoted MAVE_nsym and MAVE_ngrid. The parameters MAVE_nsym and MAVE_ngrid specify the grid density in the computation of powder molar magnetisation. The program uses Lebedev-Laikov distribution of points on the unit sphere. The parameters are integer numbers: n_{sym} and n_{grid}. The n_{sym} defines which part of the sphere is used for averaging. It takes one of the three values: 1 (half-sphere), 2 (a quarter of a sphere) or 3 (an octant of the sphere). n_{grid} takes values from 1 (the smallest grid) till 32 (the largest grid, i.e. the densest). The default is to consider integration over a half-sphere (since $M(H) = -M(-H)$): $n_{sym} = 1$ and $n_{sym} = 15$ (i.e 185 points distributed over half-sphere). In case of symmetric compounds, powder magnetisation may be averaged over a smaller part of the sphere, reducing thus the number of points for the integration. The user is responsible to choose the appropriate integration scheme. Note that the program's default is rather conservative.

Example:

```
MAVE
1  8
```

TEXP

This keyword allows computation of the magnetic susceptibility $\chi T(T)$ at experimental points. On the line below the keyword, the number of experimental points N_T is defined, and on the next N_T lines the program reads the experimental temperature (in K) and the experimental magnetic susceptibility (in $cm^3 K mol^{-1}$). The magnetic susceptibility routine will also print the standard deviation from the experiment.

```
TEXP
54
299.9901   55.27433
290.4001   55.45209
279.7746   55.43682
269.6922   55.41198
259.7195   55.39274
249.7031   55.34379
239.735    55.29292
229.7646   55.23266
219.7354   55.15352
209.7544   55.06556
...
```

HEXP

This keyword allows computation of the molar magnetisation $M_{mol}(H)$ at experimental points. On the line below the keyword, the number of experimental points N_H is defined, and on the next N_H lines the program reads the experimental field intensity (in Tesla) and the experimental magnetisation (in μ_{Bohr}). The magnetisation routine will print the standard deviation from the experiment.

```
HEXP
3  1.0  5.3  2.4  # temperature values
10 # numer of field points
0.30 4.17 1.26 2.51  # H(T) and M for each temperature
1.00 5.47 3.57 4.82
1.88 5.79 4.54 5.30
2.67 5.92 4.96 5.54
3.46 5.97 5.20 5.70
4.24 6.00 5.36 5.81
5.03 6.01 5.48 5.88
5.82 6.02 5.57 5.93
6.61 6.02 5.65 5.97
7.40 6.03 5.72 5.99
```

ZJPR

This keyword specifies the value (in cm^{-1}) of a phenomenological parameter of a mean molecular field acting on the spin of the complex (the average intermolecular exchange constant). It is used in the calculation of all magnetic properties (not for spin Hamiltonians) (Default is 0.0).

```
ZJPR
-0.02
```

XFIE

This keyword specifies the value (in T) of applied magnetic field for the computation of magnetic susceptibility by dM/dH and M/H formulas. A comparison with the usual formula (in the limit of zero applied field) is provided. (Default is 0.0). Example:

```
XFIE
0.35
```

This keyword together with the keyword `PLOT` will enable the generation of two additional plots: `XT_with_field_dM_over_dH.png` and `XT_with_field_M_over_H.png`, one for each of the two above formula used, alongside with respective `gnuplot` scripts and gnuplot datafiles.

TORQ

This keyword specifies the number of angular points for the computation of the magnetisation torque function, $\vec{\tau}_\alpha$ as function of the temperature, field strength and field orientation.

```
TORQ
55
```

The torque is computed at all temperature given by `TMAG` or `HEXP_temp` inputs. Three rotations around Cartesian axes X, Y and Z are performed.

PRLV

This keyword controls the print level.

- 2 - normal. (Default)
- 3 or larger (debug)

PLOT

Set to "true", the program generates a few plots (png or eps format) via an interface to the linux program *gnuplot*. The interface generates a datafile, a *gnuplot* script and attempts execution of the script for generation of the image. The plots are generated only if the respective function is invoked. The magnetic susceptibility, molar magnetisation and blocking barrier (`UBAR`) plots are generated. The files are named: `XT_no_field.dat`, `XT_no_field.plt`, `XT_no_field.png`, `MH.dat`, `MH.plt`, `MH.png`, `BARRIER_TME.dat`, `BARRIER_ENE.dat`, `BARRIER.plt` and `BARRIER.png`, `zeeman_energy_xxx.png` etc. All files produced by `SINGLE_ANISO` are referenced in the corresponding output section. Example:

```
PLOT
```

5.36 DeltaSCF

`DeltaSCF` is a time-independent alternative to time-dependent DFT for the description of excited electronic states in a mean-field DFT framework and has been shown to yield better results than TDDFT in the common linear-response adiabatic approximation for the topology of an avoided crossing and a conical intersection in the ethylene molecule paradigmatic for photochemistry [758], as well as for charge transfer excited states, where the absolute error in TDDFT increases rapidly with the charge transfer distance, while a balanced description is obtained with `DeltaSCF` [759].

Typically, the SCF proceedure is used to obtain the ground electronic state represented by a minimum on the energy surface with respect to the electronic orbital rotation degrees of freedom. The idea behind ΔSCF is to converge to a higher-energy solution on the electronic energy surface, often a saddle point instead, in order to find mean-field representations of excited electronic states. As it is usually not sufficient to form an initial guess close to such an excited state solution and use conventional SCF approaches, several strategies have been developed and implemented in ORCA to make the SCF converge to the target saddle point and prevent so-called variational collapse, convergence to undesired lower-energy solutions.

One simple but often effective approach was introduced by the group of Peter Gill [760] as the Maximum Overlap Method (MOM). In ORCA, we also feature the more recent PMOM from the group of Hrant Hratchian [761]. Additionally, ORCA offers an SOSCF optimization technique that leverages the limited-memory symmetric rank-1 quasi-Newton method as well as the generalized mode following approach both capable of converging to saddle point solutions systematically. The excited state initial guess can be refined by constrained optimization within the freeze-and-release SOSCF strategy. ΔSCF has also been referred to as "orbital optimized DFT for electronic excited states" [762] and "excited state mean-field theory" [763].

Figure Fig. 5.67 shows the schematic representation of a generic HOMO-LUMO excitation in ΔSCF. First, the ground state is obtained in a regular SCF calculation. Then, the occupation numbers are changed to a non-aufbau occupation reflecting the desired excitation. For a HOMO-LUMO excitation, the HOMO in one of the two spin channels is chosen to be unoccupied, while the LUMO is occupied. This non-aufbau occupation together with the ground state orbitals is used as the initial guess for a variational saddle point search yielding a ΔSCF excited state solution.

Fig. 5.67: A simple scheme of a HOMO-LUMO excitation state using ΔSCF

Using ORCA's `DeltaSCF`, we can **choose** to converge the SCF to a HOMO/LUMO excited state. The excited state is obtained by fully relaxing the orbitals and can include any contributions such as the VV10 correlation or CPCM for solvation. Due to the variational optimization of the wave function, the Hellman-Feynman holds meaning that the gradient on the positions of the nuclei is available for geometry optimizations, molecular dynamics simulations, reaction path calculations, and all the other geometry search features ORCA has to offer. Additionally, all properties ORCA can calculate with a ground state SCF density can be computed with an excited state ΔSCF density as well.

Important

ΔSCF states are represented by single-determinant wave functions within a DFT framework. As ground state DFT can struggle with multideterminantal cases, so can excited state DFT. Keep this in mind when tackling such cases!

This description is, for instance, relatively reasonable when

1. the excited state can be described by a particle-hole interaction. That is for example **NOT** the case for the HOMO-LUMO excitation of a benzene molecule or most π-π^* excited states,
2. the occupied and virtual orbitals are separated in space such that the relevant exchange integral is zero. Examples include some n-π^* states, long-range charge transfer states etc.,
3. for closed-shell doubly excited states which can be represented by a single-determinant wavefunction.

Furthermore, ΔSCF wave functions can break the symmetries of the Hamiltonian (this is true for ground and excited states). Open-shell singly excited states inherently break the spin symmetry and require spin purification [764].

5.36.1 Theory

In this section, the theory of behind the ΔSCF method is discussed.

SOSCF L-SR1 method

In ΔSCF, one searches a stationary point on the energy surface with respect to the electronic orbital rotation degrees of freedom collected in the orbital rotation matrix

$$\boldsymbol{\kappa} = \begin{pmatrix} \boldsymbol{\kappa}_{\text{oo}} = 0 & \boldsymbol{\kappa}_{\text{ov}} \\ -\boldsymbol{\kappa}_{\text{ov}}^{+} & \boldsymbol{\kappa}_{\text{vv}} = 0 \end{pmatrix} ,$$

where oo, ov, and vv represent the blocks of $\boldsymbol{\kappa}$ containing pairwise orbital rotations that mix two occupied, an occupied and a virtual, and two virtual orbitals, respectively. A stationary point is found as

$$\mathbf{C}_{\text{stat}} = \mathbf{C}_0 e^{\boldsymbol{\kappa}} ,$$

with the energy gradient being

$$\frac{\partial E}{\partial \kappa_{ij}} = \frac{2-\delta_{ij}}{2} \left[\int_0^1 dt e^{t\boldsymbol{\kappa}} \mathbf{L} e^{-t\boldsymbol{\kappa}} \right]_{ji} = \frac{2-\delta_{ij}}{2} \left[\mathbf{L} + \frac{1}{2!} [\boldsymbol{\kappa}, \mathbf{L}] + \frac{1}{3!} [\boldsymbol{\kappa}, [\boldsymbol{\kappa}, \mathbf{L}]] + ... \right]_{ji} ,$$

where the right-hand side is a special case of the Baker-Campbell-Hausdorff formula and holds because the space of $\boldsymbol{\kappa}$ forms the Lie algebra corresponding to the Lie group of its exponential. Since the gradient is always zero for all elements of $\boldsymbol{\kappa}_{\text{oo}}$ and $\boldsymbol{\kappa}_{\text{vv}}$, those blocks of $\boldsymbol{\kappa}$ are dropped as degrees of freedoms in the optimization. As $e^{\boldsymbol{\kappa}}$ is required to be unitary, $\boldsymbol{\kappa}$ is required to be anti-Hermitian, dropping several orbital rotations as degrees of freedom in the optimization.

The saddle point search is performed using the L-SR1 quasi-Newton optimizer, which is capable of forming an indefinite model Hessian, starting from an excited state initial guess. To increase the performance of the method and make it more robust, the following standard SOSCF preconditioner is used and allowed to become indefinite as well:

$$\frac{\partial^2 E}{\partial \kappa_{ij}^2} \approx 2 \left(f_j - f_i \right) \left(\epsilon_i - \epsilon_j \right) ,$$

where f_i is the occupation number of orbital i and ϵ_i its energy.

Generalized Mode Following

The generalized mode following (GMF) method [765, 766] is a more robust but also a bit more expensive saddle point search method that can be used to converge to ΔSCF excited state solutions. The method recasts the saddle point search as a minimization by modifying the gradient, such that it corresponds to a modified objective function that has a minimum where the energy function has the target saddle point. Now, a minimization can be performed to locate the target saddle point corresponding to the desired excited state, meaning the robust L-BFGS optimizer can be used instead of the L-SR1 optimizer, the preconditioner can be made positive-definite, and variational collapse to lower-energy solutions other than the targeted one becomes impossible by construction, making MOM unnecessary with GMF. For this to work, the saddle point order n of the target solution needs to be estimated at the start of the calculation, and the lowest n eigenpairs of the electronic Hessian need to be evaluated at every electronic optimization step with a finite difference Davidson method.

Once the lowest n eigenvectors v_i of electronic Hessian are evaluated at each optimization step, the modified gradient is calculated as

$$g^{\text{mod}} = g - 2 \sum_{i=1}^{n} v_i v_i^{\text{T}} g .$$

5.36.2 Freeze-and-release strategy

In tricky cases, particularly those where the excitation leads to a large amount of charge transfer, it can be advantageous to shift to a freeze-and-release SOSCF strategy [767], which converges the DeltaSCF in two steps:

```
1. A constrained optimization freezing all orbitals that are excited, i.e. those
↪orbitals
   for which the occupation numbers differ from the aufbau occupation. This step
↪reduces
   to a minimization for the unconstrained orbitals.

2. An unconstrained optimization as usual.
```

Step 1 accounts for the orbital relaxation effect due to the first-order change in the electron density after the excitation on the orbitals that are not involved in the excitation. It can be regarded as an elaborate way of generating a very good initial guess for the excited state saddle point search in step 2.

5.36.3 Input and examples

In this section, some example calculations are presented together with the input.

First Example: HOMO-LUMO Excited State of Formaldehyde

Let's begin by trying to converge to the first excited state of formaldehyde and optimize the geometry in that electronic state starting from the regular planar structure:

```
!PBE DEF2-TZVP OPT FREQ DELTASCF UHF
%SCF ALPHACONF 0,1 END
* XYZ 0 1
C      0.000000    0.000000   -0.602985
O      0.000000    0.000000    0.605394
H      0.000000    0.934673   -1.182175
H      0.000000   -0.934673   -1.182175
*
```

Besides the regular keywords such as method, basis set, `OPT` and `FREQ`, one needs to specify `DELTASCF` in the simple input and in this case `UHF` since the alpha and beta orbitals will be different due to the single excitation (for doubly-excited states `RHF` might be sufficient).

It is also necessary to add the `ALPHACONF` or `BETACONF` under the `%SCF` block. That is a minimal representation of the configuration you want to converge to. In this case, `0,1` means a HOMO/LUMO transition, where the HOMO has occupation zero and LUMO occupation one. For a HOMO-1/LUMO transition, it would be `ALPHACONF 0,1,1`. For a HOMO/LUMO+1 it would be `ALPHACONF 0,0,1` and so on. Just picture what the frontier orbital occupation should look like. Any orbital below the first zero is assumed to be occupied and any orbital above the last occupied is assumed to be empty.

After the regular startup, the ΔSCF-specific print shows:

```
-------------------------------
DELTA-SCF INITIAL CONFIGURATION
-------------------------------

Alpha: 1.00 1.00 1.00 0.00 1.00 0.00 0.00 0.00
Beta : 1.00 1.00 1.00 1.00 0.00 0.00 0.00 0.00

Hessian update                              ... L-SR1
Aufbau metric                               ... MOM
Keep initial reference                      ... true
```

Here you can follow the initial configuration and some other important things:

1. `Hessian update` refers to which method will be used for the SOSCF Hessian update. ORCA's default is `L-BFGS`, which forces the electronic Hessian to be positive definite and will always push the system down to a minimum. As we want to go to a saddle point, the `L-SR1` is set by default for `DeltaSCF`. More details on Hessian updates and their consequences at this reference from the group of Hannes Jónsson [768].
2. `Aufbau metric` is the way one measures the "overlap" between the actual and reference wave functions (more details in [761]). It is `MOM` by default, but can be also set to `%SCF PMOM TRUE END` to use `PMOM`.
3. `Keep initial reference TRUE` means we will always try to keep the initial reference state defined after the guess phase. That is sometimes called `IMOM` in the literature [769]. If `%SCF KEEPINITIALREF FALSE END` is set, it is always the last SCF iteration that is taken as reference.

Important

Here we are using the orbitals of the `PMODEL` guess because it is trivial. In general, we recommend always starting the SCF by reading the orbitals of a previously converged ground state SCF! Please check *Restarting SCF Calculations* for more info on that.

This calculation trivially converges in 12 steps:

```
----------------------------------------D-I-I-S-----------------------------------
--------
Iteration    Energy (Eh)           Delta-E    RMSDP     MaxDP     DIISErr   Damp
Time(sec)
-------------------------------------------------------------------------------
--------
               ***  Starting incremental Fock matrix formation  ***
MOM changed the orbital occupation numbers
    1    -114.2533654537761123     0.00e+00  2.06e-03  6.64e-02  7.14e-02  0.700
0.1
MOM changed the orbital occupation numbers
    2    -114.2675698961360382    -1.42e-02  4.71e-04  1.46e-02  3.08e-02  0.700
0.1
                              ***Turning on AO-DIIS***
MOM changed the orbital occupation numbers
    3    -114.2754325617175226    -7.86e-03  1.98e-04  3.98e-03  2.21e-02  0.700
0.1
MOM changed the orbital occupation numbers
    4    -114.2806616906060100    -5.23e-03  4.98e-04  6.40e-03  1.60e-02  0.000
0.1
MOM changed the orbital occupation numbers
                             *** Initializing SOSCF ***
---------------------------------------S-O-S-C-F-----------------------------------
---
Iteration    Energy (Eh)           Delta-E    RMSDP     MaxDP     MaxGrad
Time(sec)
-------------------------------------------------------------------------------
---
    5    -114.2932665018211225    -1.26e-02  7.19e-05  1.55e-03  2.33e-03     0.1
               *** Restarting incremental Fock matrix formation ***
                            *** Restarting Hessian update ***
    6    -114.2932913947584979    -2.49e-05  5.82e-05  1.33e-03  1.11e-03     0.1
    7    -114.2932671856877818     2.42e-05  2.54e-05  5.00e-04  2.58e-03     0.1
    8    -114.2932914986330530    -2.43e-05  1.76e-05  3.20e-04  1.02e-03     0.1
    9    -114.2932961472105404    -4.65e-06  2.13e-06  6.94e-05  6.42e-05     0.1
   10    -114.2932961779292640    -3.07e-08  2.99e-05  1.10e-03  8.27e-06     0.1
   11    -114.2932921911087050     3.99e-06  3.05e-05  1.12e-03  5.44e-04     0.1
   12    -114.2932961794015654    -3.99e-06  2.45e-08  5.31e-07  2.39e-07     0.1
                 *** Gradient check signals convergence ***
```

Note

The statement `MOM changed the orbital occupation numbers` is normal, it is just printing what it is doing. Nothing to worry about.

and one can see from the spin contamination, that this is indeed an open-shell singlet:

```
----------------------
UHF SPIN CONTAMINATION
----------------------

Warning: in a DFT calculation there is little theoretical justification to
         calculate <S**2> as in Hartree-Fock theory. We will do it anyways
         but you should keep in mind that the values have only limited relevance

Expectation value of <S**2>     :     1.006910
Ideal value S*(S+1) for S=0.0   :     0.000000
Deviation                       :     1.006910
```

The gradient is then computed, and the geometry is optimized until convergence. Finally the frequencies show this is actually not a minimum, but a saddle point on the geometry space!

```
-----------------------
VIBRATIONAL FREQUENCIES
-----------------------

Scaling factor for frequencies =  1.000000000  (already applied!)

     0:       0.00 cm**-1
     1:       0.00 cm**-1
     2:       0.00 cm**-1
     3:       0.00 cm**-1
     4:       0.00 cm**-1
     5:       0.00 cm**-1
     6:    -591.31 cm**-1 ***imaginary mode***
     7:     799.56 cm**-1
     8:    1228.37 cm**-1
     9:    1271.29 cm**-1
    10:    2971.77 cm**-1
    11:    3067.96 cm**-1
```

The reason is: the HOMO/LUMO transition on formaldehyde populates the π^* LUMO, thus breaking the double bound and making the carbon atom pyramidal. If one starts from a slightly distorted structure, it then converges to the actual geometry minimum. Starting from a pyramidal structure:

```
!wB97M-D4 DEF2-TZVP DELTASCF UHF OPT FREQ
%SCF ALPHACONF 0,1 END
* XYZ 0 1
C          0.00000        0.00000       -0.60298
O         -0.74131       -0.10909        0.45746
H          0.00000        0.93467       -1.18217
H          0.00000       -0.93467       -1.18217
*
```

now converges to a minimum, as shown by the absence of negative frequencies:

```
-----------------------
VIBRATIONAL FREQUENCIES
-----------------------

Scaling factor for frequencies =  1.000000000  (already applied!)
```

(continues on next page)

(continued from previous page)

```
    0:       0.00 cm**-1
    1:       0.00 cm**-1
    2:       0.00 cm**-1
    3:       0.00 cm**-1
    4:       0.00 cm**-1
    5:       0.00 cm**-1
    6:     713.25 cm**-1
    7:     826.75 cm**-1
    8:    1206.57 cm**-1
    9:    1292.70 cm**-1
   10:    2826.06 cm**-1
   11:    2893.16 cm**-1
```

The final geometry is surprisingly accurate for the gas phase formaldehyde, too! We will also show the results here using the range-corrected, meta-GGA hybrid `wB97M-D4` and the double-hybrid DFT `!B2PLYP DEF2-QZVPP AUTOAUX`, just to show that MP2 also works.

Table 5.23: Geometry of gas phase formaldehyde versus the DeltaSCF results, in Angstroem and degrees.

parameter	exp.	PBE	wB97M-D4	B2PLYP
r(C-O)	1.323	1.300	1.310	1.311
r(C-H)	1.098	1.113	1.093	1.090
∠HCH	118.8	115.0	117.0	116.8
∠OOP	34.0	41.1	36.5	37.0

- exp. taken from [760].

Important

We are using the default ground state SCF algorithm here, `AO-DIIS` + `SOSCF` because this is relatively simple. In general and for more complicated cases, we suggest using the approximate second-order method, to avoid escaping back to the ground state with `!NODIIS NOTRAH`, which is the default for ΔSCF.

Important

Do **NOT** combine ΔSCF wavefunctions with CCSD, or any such method with single excitations. It requires a speciallized version of CC which we don't have yet.

Important

When running the same calculation above with `wB97M-D4`, there will **not** be a virtual orbital between the alpha HOMO-1 and the HOMO (so no negative HOMO-LUMO gap). There is nothing wrong here, it just optimized the orbitals to the excited state such that the occupation numbers correspond to an aufbau occupation. Hybrid functionals tend to lower the energy of the occupied orbitals with respect to the virtual orbitals.

The energy is still higher than the non-ΔSCF solution and if you plot the orbitals you will see that the alpha HOMO is now a π orbital instead of an n. The eigenvalues of the electronic Hessian indicate that this solution is a first-order saddle point.

Core-ionized States

Another big advantage of the ΔSCF is the possibility to converge to core-excited and/or core-ionized states. We have a simple keyword to remove electrons from any orbital, even the deep core ones:

```
%SCF IONIZEALPHA 2 END
```

and the electron from orbital number two will be removed. `IONIZEBETA` works for beta orbitals. One can start from an anion UHF electronic structure obtained by adding one extra electron and remove a core electron like this to obtain core-excited states too. Geometry optimization, EPR, and even TDDFT calculations are all valid for these states.

As an example, the input below will ionize the 1s electron from a water molecule, which corresponds to MO 0 here:

```
!PBE0 DEF2-TZVP DELTASCF NODIIS UHF
%SCF IONIZEALPHA 0 END
* xyz 0 1
  O      2.127880   -0.361920    0.104770
  H      3.117210   -0.387460    0.070360
  H      1.838520   -0.926280   -0.655730
*
```

Note

If the orbital is not localized over a single atom, one might need to localized them first!

and one can see from the results that is exactly what it converges to.

```
----------------
ORBITAL ENERGIES
----------------
                  SPIN UP ORBITALS
  NO   OCC          E(Eh)            E(eV)
   0   0.0000     -20.108086      -547.1688
   1   1.0000      -1.567415       -42.6515
   2   1.0000      -1.057257       -28.7694
   3   1.0000      -0.939315       -25.5601
   4   1.0000      -0.882871       -24.0241
   5   0.0000      -0.304254        -8.2792
   6   0.0000      -0.239285        -6.5113
   7   0.0000       0.044344         1.2066
        (...)
```

Note

ORCA automatically adjusts charge and multiplicity here. **The input should contain those from the reference system!**

Here are some examples of binding energies for 1s electrons. The atom from where it was removed is highlighted in bold:

Table 5.24: Binding energies from 1s electrons found by DeltaSCF using wB97M-V/DEF2-TZVPP, in eV.

1s ionization	exp.	wB97M-V
H2**O**	539.82	541.17
CO2	297.69	299.10
NH3	405.56	406.82
CH3C**N**	405.64	406.92

- exp. taken from [770]

Diabatic Couplings

ΔSCF is also a quite accurate method to obtained diabatic couplings, which can later be used in Markus theory to compute electron transfer rates. These can be computed by calculating the energy difference between electron transfer states and using the Generalized Mulliken-Hush Approach (GMH). For more details please check for example this paper from the group of Blumberger [771].

There is not enough space to go through the details here, but one can get these diabatic coupling from essentially one regular SCF for the ground state + a ΔSCF for the excited state. For symmetric systems, this is trivial:

$$2|H_{ab}| = \Delta E_{12}$$

where states a and b are diabatic states, and ΔE_{12} is the energy difference between adiabatic states 1 and 2 (which are obtained via SCF solution). Here is an example of the diabatic couplings obtained for a benzene dimer, obtained by starting from the ground state cation and exciting the beta electron with:

```
%SCF BETACONF 0,1 END
```

Table 5.25: Diabatic couplings found by ΔSCF using wB97M-V/DEF2-TZVPP, in meV.

Distance in Ang	MRCI+Q	wB97M-V	TD-DFT
3.5	435.2	473.1	593
4.0	214.3	236.7	374
4.5	104.0	115.9	267.5
5.0	51.70	56.7	218.5

- MRCI+Q taken from [771]

Generalized Mode Following

For an example GMF calculation, we return to the formaldehyde molecule, this time targeting a HOMO-LUMO+1 excitation to keep it interesting and add the simple input keyword `GMF`:

```
!WB97X-D4 DEF2-SVP UKS DELTASCF GMF
%SCF ALPHACONF 0,0,1 END
* XYZ 0 1
O         -3.54418        1.61112        0.04891
C         -4.12047        0.61074       -0.35955
H         -3.36418       -0.41948       -0.79627
H         -4.84699        0.73407       -1.42962
*
```

If not provided in the input, the target saddle point order is estimated either by using the orbital energies or by evaluating the number of negative eigenvalues of the electronic Hessian. In the former case, the target saddle point order is approximated as the number of pairs of an occupied and a virtual orbital where the occupied orbital has a larger energy than the virtual orbital at the excited state initial guess after the excitation has been performed. In our example, this yields a saddle point order of 2, as is estimated by default and printed together with some other default input parameters in the `DeltaSCF` block:

```
-------------------------------
DELTA-SCF INITIAL CONFIGURATION
-------------------------------

Alpha: 1.00 1.00 1.00 0.00 0.00 1.00 0.00 0.00 0.00
Beta : 1.00 1.00 1.00 1.00 0.00 0.00 0.00 0.00 0.00
```

(continues on next page)

(continued from previous page)

```
Hessian update                              ... L-BFGS/GMF
Generalized mode following:
 Target saddle point order                  ... 2
 Finite difference generalized Davidson method:
  Finite difference mode                    ... Forward
  Finite difference step size               ... 1.000e-03
  Max. size of Krylov subspace              ... 20
  Convergence tolerance                     ... 1.000e-02
  Max. number of iterations                 ... 100
Aufbau metric                               ... Fixed occupations
```

Before every optimization step, the lowest 2 (same as target saddle point order) eigenpairs of the electronic Hessian are evaluated, and once converged below the tolerance, the eigenvalues are printed, e.g. for the first optimization step:

```
Solving for 2 Hessian eigenvectors
    It.    Root MAX Err.:          1          2
    1                       6.520e-02  6.276e-02
    2                       3.193e-02  2.817e-02
    3                       1.511e-02  9.423e-03
    4                       3.889e-03  9.417e-03
    Eigenvalues:           -3.316e-01 -2.404e-01
```

After a few optimization steps, the evaluation of the Hessian eigenpairs requires no further adjustments, so that the Davidson runs converge immediately:

```
        Solving for 2 Hessian eigenvectors
            It.    Root MAX Err.:          1          2
            1                       5.165e-03  8.433e-03
            Eigenvalues:           -2.574e-01 -1.792e-01
   11    -114.1023998750192305    -1.06e-05  1.06e-03  1.96e-02  5.46e-04     1.3
        Solving for 2 Hessian eigenvectors
            It.    Root MAX Err.:          1          2
            1                       5.755e-03  8.231e-03
            Eigenvalues:           -2.575e-01 -1.791e-01
   12    -114.1024101850037624    -1.03e-05  6.43e-04  1.25e-02  3.11e-04     1.3
        Solving for 2 Hessian eigenvectors
            It.    Root MAX Err.:          1          2
            1                       6.190e-03  8.354e-03
            Eigenvalues:           -2.575e-01 -1.790e-01
   13    -114.1024129524389963    -2.77e-06  3.67e-04  7.30e-03  2.00e-04     1.3
        Solving for 2 Hessian eigenvectors
            It.    Root MAX Err.:          1          2
            1                       6.264e-03  8.286e-03
            Eigenvalues:           -2.574e-01 -1.790e-01
   14    -114.1024142211099530    -1.27e-06  8.68e-05  1.34e-03  1.46e-04     1.3
        Solving for 2 Hessian eigenvectors
            It.    Root MAX Err.:          1          2
            1                       6.396e-03  8.439e-03
            Eigenvalues:           -2.574e-01 -1.790e-01
   15    -114.1024145538739987    -3.33e-07  4.56e-05  7.31e-04  9.17e-05     1.3
                  **** Energy Check signals convergence ****
```

In this simple example, one would expect the configuration to be conserved during the optimization. This is not the case here, again, because a hybrid functional is used. The occupied orbitals are stabilized with respect to the virtual ones leading to this configuration:

```
----------------
ORBITAL ENERGIES
----------------
                 SPIN UP ORBITALS
  NO   OCC          E(Eh)            E(eV)
```

(continues on next page)

(continued from previous page)

```
   0   1.0000     -19.374322      -527.2021
   1   1.0000     -10.431285      -283.8497
   2   1.0000      -1.258409       -34.2430
   3   1.0000      -0.755807       -20.5666
   4   1.0000      -0.635639       -17.2966
   5   1.0000      -0.591213       -16.0877
   6   1.0000      -0.584445       -15.9036
   7   0.0000      -0.143277        -3.8988
   8   1.0000      -0.113247        -3.0816
   9   0.0000      -0.061224        -1.6660
  10   0.0000       0.146574         3.9885
  11   0.0000       0.251438         6.8420
  12   0.0000       0.490702        13.3527
  13   0.0000       0.537161        14.6169
  14   0.0000       0.565655        15.3923
  15   0.0000       0.624632        16.9971
  16   0.0000       0.649717        17.6797
  17   0.0000       0.710983        19.3468
  18   0.0000       0.928559        25.2674
```

The two lowest eigenvalues of the Hessian at the last optimization step are clearly negative, so this solution is a second-order saddle point, as expected and targeted in the optimization. Do not rely on the configuration to determine the saddle point order if you use hybrid functionals!

Freeze-and-release strategy

Let's take a look at a tricky excited state of the 90-degrees-twisted N-phenylpyrrole molecule. We first perform a ground state SCF calculation without `DeltaSCF` using this input:

```
!UKS aug-cc-pVDZ PBE

*xyz 0 1
C       10.13954933      10.09711395      12.38421108
C       11.34646202      10.09711395      13.07460199
C        8.93263664      10.09711395      13.07460199
C       10.13954933      10.09711395      15.16101197
C       11.34465495      10.09711395      14.46510328
C        8.93444370      10.09711395      14.46510328
C       10.13954933      11.21630306      10.16939448
C       10.13954933       8.97792484      10.16939448
C       10.13954933      10.80761594       8.85675649
C       10.13954933       9.38661196       8.85675649
N       10.13954933      10.09711395      10.96513494
H       12.26979504      10.09711395      12.51888639
H        8.00930362      10.09711395      12.51888639
H       12.27909865      10.09711395      15.00190089
H        8.00000000      10.09711395      15.00190089
H       10.13954933      10.09711395      16.23847837
H       10.13954933      12.19422790      10.60962753
H       10.13954933       8.00000000      10.60962753
H       10.13954933      11.45309491       8.00000000
H       10.13954933       8.74113299       8.00000000
*
```

This calculation yields an energy of -440.702 E_h and a dipole moment of 2.181 D. Now we perform an excited state calculation with `DeltaSCF` starting from this converged ground state solution and exciting an electron from the HOMO to the LUMO in the alpha spin channel with this input:

```
!UKS aug-cc-pVDZ PBE DeltaSCF
!SOSCF NoDIIS NoTRAH
```

(continues on next page)

(continued from previous page)

```
!MORead
!SCFCheckGrad

%moinp "gs.gbw"

%scf
AlphaConf 0,1
DoMOM false
SOSCFMaxStep 0.1
end

*xyz 0 1
C       10.13954933      10.09711395      12.38421108
C       11.34646202      10.09711395      13.07460199
C        8.93263664      10.09711395      13.07460199
C       10.13954933      10.09711395      15.16101197
C       11.34465495      10.09711395      14.46510328
C        8.93444370      10.09711395      14.46510328
C       10.13954933      11.21630306      10.16939448
C       10.13954933       8.97792484      10.16939448
C       10.13954933      10.80761594       8.85675649
C       10.13954933       9.38661196       8.85675649
N       10.13954933      10.09711395      10.96513494
H       12.26979504      10.09711395      12.51888639
H        8.00930362      10.09711395      12.51888639
H       12.27909865      10.09711395      15.00190089
H        8.00000000      10.09711395      15.00190089
H       10.13954933      10.09711395      16.23847837
H       10.13954933      12.19422790      10.60962753
H       10.13954933       8.00000000      10.60962753
H       10.13954933      11.45309491       8.00000000
H       10.13954933       8.74113299       8.00000000
*
```

This simple-seeming excitation leads to a large charge transfer from the phenyl group to the pyrrole group which makes this excited state tricky to converge. To this end, we reduce the maximum step size of the SOSCF optimization from 0.2 (default) to 0.1 with `SOSCFMaxStep 0.1` and deactivate MOM with `DoMOM false` since MOM tends to become counterproductive for cases with heavy charge transfer. We also use only SOSCF and deactivate DIIS and TRAH using the simple input `!SOSCF NoDIIS NoTRAH` and force a convergence check for the SOSCF gradient with `!SCFCheckGrad`. The latter amounts to personal preference.

This calculation converges to a solution with an excitation energy of 189 mE_h and a dipole moment difference to the ground state of 4.958 D, which are both smaller than expected for this heavy charge transfer state. This is known as a variational collapse, i.e. the calculation converged to a lower-energy solution than desired.

To counteract this variational collapse, we use the freeze-and-release SOSCF approach by adding `FreezeAndRelease` (or `FRSOSCF`) to the simple input line:

```
!UKS aug-cc-pVDZ PBE DeltaSCF FreezeAndRelease
!MORead
!SCFCheckGrad

%moinp "gs.gbw"

%scf
AlphaConf 0,1
DoMOM 0
SOSCFConvFactor 500
SOSCFMaxStep 0.1
end
```

(continues on next page)

(continued from previous page)

```
*xyz 0 1
C       10.13954933      10.09711395      12.38421108
C       11.34646202      10.09711395      13.07460199
C        8.93263664      10.09711395      13.07460199
C       10.13954933      10.09711395      15.16101197
C       11.34465495      10.09711395      14.46510328
C        8.93444370      10.09711395      14.46510328
C       10.13954933      11.21630306      10.16939448
C       10.13954933       8.97792484      10.16939448
C       10.13954933      10.80761594       8.85675649
C       10.13954933       9.38661196       8.85675649
N       10.13954933      10.09711395      10.96513494
H       12.26979504      10.09711395      12.51888639
H        8.00930362      10.09711395      12.51888639
H       12.27909865      10.09711395      15.00190089
H        8.00000000      10.09711395      15.00190089
H       10.13954933      10.09711395      16.23847837
H       10.13954933      12.19422790      10.60962753
H       10.13954933       8.00000000      10.60962753
H       10.13954933      11.45309491       8.00000000
H       10.13954933       8.74113299       8.00000000
*
```

`FreezeAndRelease` only works for SOSCF, so `SOSCF`, `NoDIIS`, and `NoTRAH` are implicite and do not need to be specified. We also apply a factor of 500 that the convergence criteria of the constrained minimization are multiplied by since the constrained minimization does not need to be converged as tightly as the subsequent unconstrained optimization. This strategy saves some iterations in the constrained part of the calculation without compromising the final result.

This strategy yields an excitation energy of 194 mE_h, a bit higher than the previous one, and a dipole moment difference to the ground state of 9.669 D. This is the desired result.

Typically, the freeze-and-release approach takes a few more iterations than the regular SOSCF, but it avoids variational collapse much more reliably. In the presented case, the regular SOSCF converges in a total of 38 energy/gradient evaluations, while the freeze-and-release strategy takes only 21 energy/gradient evaluations.

Note

The statement `Scaled step length from x to 0.100000` is indicating that a relatively large step was about to be taken. Scaling it down typically results in more robust but less efficient convergence. It can be adjusted to the challenges of the system under investigation.

The freeze-and-release strategy can be combine with GMF yielding an even more robust strategy. In this case, the target saddle point order is estimated at the solution of the constrained minimization step since the quality of the estimate increases then. For hybrid functionals, the default estimate is obtained in the first Davidson run. Otherwise, the orbital energies are used.

5.36.4 Full keyword list

The simple input keywords related to ΔSCF are collected in tab. Table 5.26.

Table 5.26: Simple input keywords for ΔSCF.

Keyword	Description
DeltaSCF	Activates ΔSCF
FreezeAndRelease or FRSOSCF	Activates freeze-and-release strategy
GMF	Activates generalized mode following
SCFLBFGS	Sets SOSCF quasi-Newton method to L-BFGS
SCFLSR1	Sets SOSCF quasi-Newton method to L-SR1

The input keywords of the %SCF block related to ΔSCF are collected in tab. Table 5.27.

Table 5.27: %scf block input keywords for ΔSCF with defaults in pare

Keyword	Options	Description
SOSCFMaxStep (0.2)	<real>	Maximum step size
DeltaSCFFromGS (true)	true/false	Start DeltaSCF from an input converged ground state solution or as
SOSCFBlockDiag (false)	true/false	Perform a diagonalization of the occupied and virtual orbital blocks
DoMOM (true)	true/false	Use the maximum overlap method?
KeepInitialRef (true)	true/false	Always keep initial reference: IMOM?
PMOM (false)	true/false	Use the PMOM metric instead of the regular MOM?
AlphaConf and BetaConf		Define the occupation of the frontier orbitals in alpha and beta spin
	0,1	HOMO→LUMO excitation
	0,0,1	HOMO→LUMO+1 excitation
	0,1,1	HOMO-1→LUMO excitation
	0,2	Double HOMO→LUMO excitation (only valid for RHF and Alph
IonizeAlpha and IonizeBeta	<int>	Remove electron from specified MOs in alpha and beta spin channe
SOSCFHessUp (LSR1)		SOSCF quasi-Newton method for Hessian update
	LSR1	Limited-memory symmetric rank-1 update
	LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno update
	LPOWELL	Limited-memory Powell update
	LBOFILL	Limited-memory Bofill update, a combination of L-SR1 and L-Pow
SOSCFConstraints (false)	true/false	Activate freeze-and-release SOSCF?
SOSCFConstrainedMaxStep (0.2)	<real>	Maximum step size for the constrained minimization
SOSCFConvFactor (1)	<real>	Factor to multiply convergence criteria with in the constrained mini
SOSCFConstrainedHessUp (LBFGS)		Hessian update for constrained minimization
	LSR1	Limited-memory symmetric rank-1 update
	LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno update
	LPOWELL	Limited-memory Powell update
	LBOFILL	Limited-memory Bofill update, a combination of L-SR1 and L-Pow
SOSCFWriteConstrainedGBW (true)	true/false	Write a GBW file for the constrained solution
SOSCFGMF (false)	true/false	Use generalized mode following?
SOSCFDavidsonMaxIt (100)	<int>	Maximum number of Davidson iterations
SOSCFDavidsonTolR (0.01)	<real>	Davidson convergence tolerance for the maximum component of ea
SOSCFDavidsonMaxRed (20)	<int>	Davidson maximum size of the Krylov subspace per target eigenvec
SOSCFDavidsonFDMode (Forward)		Davidson finite difference stencil to use
	Forward	One displaced gradient evaluation, linear accuracy in finite differenc
	Central	Two displaced gradient evaluations, quadratic accuracy in finite diff
SOSCFDavidsonFDStep (0.001)	<real>	Davidson finite difference step size
SOSCFSPO	<int>	GMF Optional user-defined target saddle point order
SOSCFSPOEst (Auto)		GMF Target saddle point order estimate
	Auto	Let the algorithm determine what is best
	Excitation	Use the excitation configuration
	HDiag	Use the diagonal Hessian approximation (only different from Exci
	Davidson	Use number of negative eigenvalues in first Davidson run (targets m

5.37 Extended Transition State with Natural Orbitals for Chemical Valence (ETS-NOCV/EDA-NOCV)

Caution

The NOCV implementation in ORCA 5.0 has been rewritten. Consequently, the generation of NOCVs no longer requires the user to run single-point calculations for the molecular fragments beforehand. This way of producing the NOCVs is deprecated.

In ORCA chemical bonds can be analyzed using the EDA-NOCV framework. This framework combines the Extended Transition State (ETS) energy decomposition method with the Natural Orbitals for Chemical Valence (NOCV) to analyze the electron density rearrangement taking place upon bond formation [772]. The ETS method decomposes the energy associated with bond formation into four components:

$$\Delta E_{\text{int}} = \Delta E_{\text{elstat}} + \Delta E_{\text{Pauli}} + \Delta E_{\text{orb}} + \Delta E_{\text{disp}}$$

This represents the "instantaneous" interaction between two non-interacting molecular fragments that leads to the bond formation process. The molecular fragments are established by dividing the molecule into two fragments according to the bond that is being studied. By non-interacting, it is meant that the molecular fragments are considered to be at an infinite distance, and the electron density of each fragment is kept frozen. The first energy term corresponds to the electrostatic interaction that the two non-interacting molecular fragments experience when they are placed in the positions they would occupy in the molecule. The second term contains the total "exchange repulsion" due to the Pauli exclusion principle and the change in the exchange-correlation energy of the fragments when brought together.

$$\Delta E_{Pauli} = \Delta \tilde{E}_{Pauli} + \Delta E^0_{XC}$$

It is important to highlight that the value reported in the EDA/ETS decomposition as `Pauli Energy` corresponds to $\tilde{E}_{Pauli}$.

The third term of the ETS decomposition corresponds to the energy associated with the interaction between the occupied and unoccupied orbitals of one molecular fragment and the other. Thus, it is associated with the change in electron density during bond formation. Finally, the last term represents a "dispersion" contribution to the interaction and it is present only when semiclassical correction schemes (e.g., Grimme's D4 correction) are used.

The EDA-NOCV method decomposes this term into contributions associated with pairs of orbitals. These orbitals are obtained by diagonalizing the difference density matrix between the molecule and the "supermolecule" resulting from the non-interacting molecular fragments.

$$\Delta E_{orb} = \sum_{k=1}^{N/2} \Delta E_k^{orb}$$

5.37.1 Running an EDA-NOCV analysis in ORCA

Note

The current implementation of the EDA-NOCV method in ORCA 6.1 supports mean-field methods for both open-shell and closed-shell calculations.

To perform an EDA-NOCV analysis, three key components must be specified:

1. **The type of calculation**: this defines the overall computational framework and determines whether the analysis will be conducted using restricted or unrestricted mean-field methods.
2. **The method used for each fragment**: this includes the choice of basis sets and exchange-correlation functionals or other theoretical methods applied to each fragment individually. Typically, each fragment should be treated at the same level of theory as the adduct.

3. **The atoms that correspond to each fragment**: this requires defining which atoms belong to each fragment in the molecular system, allowing ORCA to partition the molecule accordingly.

The EDA-NOCV analysis is performed using the `EDA` keyword in the simple input line. In the following example, we have the input file for an EDA-NOCV analysis of the bond between the two carbon atoms in an ethane molecule. The new implementation allows the study of this system by considering two methyl groups as molecular fragments, with opposite spins for the unpaired electrons.

```
#
# Example of an EDA-NOCV calculation for the CH3-CH3 system with triplet fragments
#

!BP86 TZVP EDA        #EDA keyword

%EDA                 #EDA block
  FRAG1 "BP86 TZVP"  #Method for fragment 1
  FRAG2 "BP86 TZVP"  #Method for fragment 2
  FRAG1_C 0          #Charge of fragment 1
  FRAG1_M 2          #Multiplicity of fragment 1
  FRAG1_SF TRUE      #Flip spin option for fragment 1
  FRAG2_C 0          #Charge of fragment 2
  FRAG2_M 2          #Multiplicity of fragment 2
END

*xyz 0 1
 C  0.019664  -0.034069   0.009101
 H  0.039672  -0.069395   1.109620
 H  1.063205  -0.065727  -0.341092
 H -0.474230  -0.953693  -0.341621
 C -0.703210   1.217999  -0.497874
 H -0.723753   1.252869  -1.598316
 H -1.746567   1.250049  -0.147169
 H -0.208833   2.137544  -0.147653
*

%Frag              #Fragment assignment using %frag block
  Definition
    1 {0:3} end
    2 {4:7} end
  end
end
```

All the parameters associated with the EDA-NOCV calculation can be specified within the `%EDA` block. The method used for each fragment can be specified in two ways: by using keywords or by passing a file. The first method is done by using the `FRAG1` and `FRAG2` keywords for fragments 1 and 2, respectively. In quotes, you must specify the level of theory and basis set used for each fragment, as well as any other keyword options. It is standard to use the same level of theory for each fragment and for the molecule, but it is also possible to specify different ones.

Caution

The basis set used for each molecular fragment and the complete molecule must be consistent. Each molecular fragment may use a different basis set, but this must also be specified for the corresponding atoms/fragments in the molecule calculation using the `%basis` block. For example:

```
!B3LYP EDA

%basis
  FragBasis 1 "TZVP"
  FragBasis 2 "def2-SVP"
end

%EDA
```

```
  FRAG1 "B3LYP TZVP"
  FRAG2 "B3LYP def2-SVP"
 end
 .
 .
 .
```

If additional parameters are required for the calculations of the fragments it is possible to pass them in a separate file by using the keywords FRAG1_METHODFILE and FRAG2_METHODFILE.

```
%EDA
  FRAG1_METHODFILE "method.txt"
  FRAG2_METHODFILE "method.txt"
END
```

This option should be used especially when a method cannot be specified by using only keywords, but a method block must be used. For example, to specify unsupported solvents in implicit solvation methods, such as C-PCM. In this case, the method file would look like:

```
BP86 TZVP

%CPCM            #CPCM block
  epsilon 80          #Dielectric constant
  refrac  1.0         #Refractive index
END
```

Note

The "!" symbol is not required when specifying methods for each fragment. Neither when using the FRAG1 and FRAG2 keywords in the %EDA block nor in the method file when using FRAG1_METHODFILE and FRAG2_METHODFILE.

The current implementation allows for charged and open-shell fragments. To specify the charge of each fragment, the keywords FRAG1_C and FRAG2_C must be used. Similarly, the keywords FRAG1_M and FRAG2_M can be used to specify the multiplicity of each fragment. By default, each fragment is considered to be neutral and closed-shell unless specified otherwise. The total charge of the fragments must add up to the charge of the molecule.

The new implementation also allows for specifying the spin of the unpaired electrons in one of the fragments. For example, in studying the CH_3-CH_3 system, the previous ORCA implementation only allowed the use of charged closed-shell fragments, CH_3^{+1} and CH_3^{-1}. Now, it is possible to consider neutral open-shell methyl groups as molecular fragments. By default, the unpaired electrons in an open-shell system are considered spin-up electrons in ORCA, but this can be changed by using the keywords FRAG1_SF and FRAG2_SF. This adjustment allows for a reduction in Pauli repulsion between the two methyl fragments.

The molecular fragments can be specified using the %Frag block or by indicating, next to each atom, to which molecular fragment they belong. This is done by placing the number 1 for the first fragment and the number 2 for the second fragment in parentheses next to each atom.

```
#
# Example of an EDA-NOCV calculation for Li-NH3 bonding
#
!BP86 TZVP EDA

%EDA
  FRAG1 "BP86 TZVP"
  FRAG2 "BP86 TZVP"
  FRAG1_M 1
  FRAG1_C 1
```

(continues on next page)

(continued from previous page)

```
END

*xyz 1 1
Li(1)     -0.212072      1.951867      0.003088
N (2)     -0.424027     -0.022414     -0.010889
H (2)     -0.914047     -0.381589     -0.841562
H (2)     -0.958117     -0.388240      0.789150
H (2)      0.468043     -0.535155      0.011083
*
```

It is important to note that when using the `%Frag` block, the indices of the atoms start from 0. For example, in the previous input for Li^+-NH_3 the fragments can alternatively be specified as:

```
%Frag            #Fragment assignment using frag block
  Definition
    1 {0} end
    2 {1:4} end
  end
end
```

5.37.2 EDA-NOCV Output File

The EDA-NOCV output is divided into two sections. The first section corresponds to the ETS Energy Decomposition Analysis. In this section, each energy term of the ETS decomposition method is printed in Hartree and kcal/mol units. Additionally, if dispersion corrections are used in the calculation, the difference between the dispersion correction for the molecule and that for the molecular fragments is printed. Here is the EDA-NOCV results for the Li^+-NH_3 input shown in the previous section:

```
-------------------------------------------------------------------------------
                         Energy Decomposition Analysis
-------------------------------------------------------------------------------
     Energy Term                      Hartree         Kcal/mol
-------------------------------------------------------------------------------
     Bond Energy                   -0.0606951779       -38.09
     Orbital Energy                -0.0181138301       -11.37
     Electrostatic Energy          -0.0706162390       -44.31
     Pauli Energy                   0.0311314946        19.54
     Delta E^0(XC)                 -0.0030956196        -1.94
     Delta Dispersion               0.0000000000         0.00
```

The second part corresponds to the orbital energy decomposition in terms of the NOCVs. In this section, the positive and negative eigenvalues associated with each NOCV pair are printed. The orbital energy contribution for each NOCV pair is given in kcal/mol. Additionally, each contribution is further divided into kinetic and potential energy components. The sum of all orbital contributions must match the total shown in the Energy Decomposition section.

```
-------------------------------------------------------------------------------
                                NOCV analysis
-------------------------------------------------------------------------------
  negative eigen.    positive eigen.    DE_k          DT_k        DV_k
         (e)                (e)       (Kcal/mol)    (Kcal/mol)    (Kcal/mol)
-------------------------------------------------------------------------------
    -0.1972432025      0.1972432025     -7.50      -104.45        96.96
    -0.0804518243      0.0804518243     -1.84        37.57       -39.41
    -0.0804301830      0.0804301830     -1.84        37.56       -39.40
    -0.0193872139      0.0193872139     -0.16         7.73        -7.89
    -0.0062264967      0.0062264967     -0.03       -14.21        14.19
    -0.0006108673      0.0006108673     -0.00         1.68        -1.68

                             Sum_k:    -11.37       -34.13        22.76
```

In the case that an open-shell system, or molecular fragment is used, the NOCV section is divided into two parts. One for the spin-up (Alpha) and other for the spin down (Beta) NOCV pairs. The Energy Decomposition section remains the same. Here is the EDA-NOCV results for the CH_3^{+1} and CH_3^{-1} input shown in the previous section:

```
-------------------------------------------------------------------------------
                        Energy Decomposition Analysis
-------------------------------------------------------------------------------
     Energy Term                      Hartree         Kcal/mol
-------------------------------------------------------------------------------
     Bond Energy                   -0.1776418532      -111.47
     Orbital Energy                -0.2616245160      -164.17
     Electrostatic Energy          -0.2104404294      -132.05
     Pauli Energy                   0.4120105303       258.54
     Delta E^0(XC)                 -0.1175097352       -73.74
     Delta Dispersion               0.0000000000         0.00


-------------------------------------------------------------------------------
                             NOCV analysis  Alpha
-------------------------------------------------------------------------------
  negative eigen.    positive eigen.    DE_k          DT_k        DV_k
         (e)                (e)      (Kcal/mol)   (Kcal/mol)   (Kcal/mol)
-------------------------------------------------------------------------------
    -0.4750327845      0.4750327845    -75.83      -124.63        48.80
    -0.0599933854      0.0599933854     -1.69         8.51       -10.19
    -0.0599917403      0.0599917403     -1.69         8.51       -10.20
    -0.0510118074      0.0510118074     -0.85       -32.69        31.84
    -0.0510053349      0.0510053349     -0.85       -32.68        31.83
    -0.0412188369      0.0412188369     -0.92         9.08       -10.00
    -0.0252608236      0.0252608236     -0.28       -19.31        19.04
    -0.0006887548      0.0006887548     -0.00        -6.18         6.18
    -0.0003325354      0.0003325354      0.00         4.22        -4.21

                       Sum_k Alpha:    -82.09      -185.17       103.09

-------------------------------------------------------------------------------
                             NOCV analysis  Beta
-------------------------------------------------------------------------------
  negative eigen.    positive eigen.    DE_k          DT_k        DV_k
         (e)                (e)      (Kcal/mol)   (Kcal/mol)   (Kcal/mol)
-------------------------------------------------------------------------------
    -0.4750299362      0.4750299362    -75.83      -124.66        48.84
    -0.0599969082      0.0599969082     -1.69         8.47       -10.16
    -0.0599919936      0.0599919936     -1.69         8.50       -10.18
    -0.0510107218      0.0510107218     -0.85       -32.65        31.80
    -0.0510066746      0.0510066746     -0.85       -32.66        31.82
    -0.0412230078      0.0412230078     -0.92         9.10       -10.01
    -0.0252605237      0.0252605237     -0.28       -19.31        19.04
    -0.0006887767      0.0006887767     -0.00        -6.19         6.18
    -0.0003325489      0.0003325489      0.00         4.21        -4.21

                       Sum_k  Beta:    -82.09      -185.20       103.11
```

Additionally, the NOCV orbitals are saved in a file named `base_name.NOCV.gbw`. These orbitals can be visualized with standard visualization programs such as Molden or Avogadro.

5.37.3 Visualization of deformation density contributions using `orca_plot`

The deformation density can be expressed as a set of contributions associated with the paired NOCVs: $\Delta\rho(r) = \sum_{k=1}^{n/2} v_k \left[-\psi_{-k}^2 + \psi_k^2\right] = \sum_{k=1}^{n/2} \Delta\rho_k(r)$

The new ORCA implementation allows for the visualization of these deformation density contributions. These densities are saved in the density container generated during the EDA-NOCV analysis. It is possible to visualize these densities by converting them into a readable format for standard visualization programs using `orca_plot`. To do this, the following command needs to be used:

```
orca_plot gbwfilename -i
```

This will show a set of different options to generate the corresponding files.

```
       1 - Enter type of plot
       2 - Enter no of orbital to plot
       3 - Enter operator of orbital (0=alpha,1=beta)
       4 - Enter number of grid intervals
       5 - Select output file format
       6 - Plot CIS/TD-DFT difference densities
       7 - Plot CIS/TD-DFT transition densities
       8 - Set AO(=1) vs MO(=0) to plot
       9 - List all available densities
      10 - Perform Density Algebraic Operations

      11 - Generate the plot
      12 - exit this program
```

Option 9 will list all the densities saved during the EDA-NOCV calculation. The densities corresponding to the NOCV pairs are denoted by `base_name.NOCV0_0.tmp`, where "base_name" is the name of the input file, and the number before the `.tmp` extension corresponds to the contribution to the deformation density. The first five density contributions are saved, so this number ranges from 0 to 4. In the case of an open-shell calculation, the first five spin-up and spin-down contributions are saved. They are labeled `.NOCV0` for spin-up and `.NOCV1` for spin-down contributions.

```
Index:                                                    Name of Density
-----------------------------------------------------------------------
    0:                                                        base_name.scfp
    1:                                                        base_name.scfr
    2:                                                     base_name.P0.tmp
    3:                                                     base_name.P1.tmp
    4:                                                base_name.NOCV0_0.tmp
    5:                                                base_name.NOCV0_1.tmp
    6:                                                base_name.NOCV0_2.tmp
    7:                                                base_name.NOCV0_3.tmp
    8:                                                base_name.NOCV0_4.tmp
    9:                                                base_name.NOCV1_0.tmp
   10:                                                base_name.NOCV1_1.tmp
   11:                                                base_name.NOCV1_2.tmp
   12:                                                base_name.NOCV1_3.tmp
   13:                                                base_name.NOCV1_4.tmp
```

To generate the plot, first select option `1 - Enter type of plot` from the main menu. Then a secondary menu will be open:

```
     1 -   molecular orbitals
     2 -   (scf) electron density          ......  (scfp   )  => AVAILABLE
     3 -   (scf) spin density              ......  (scfr   )  => AVAILABLE
     4 -   natural orbitals
     .
     .
     .
```

Here, we select the second option `2 - (scf) electron density`. Then it will ask if `base_name.scfp` is the density we want to use or not. This corresponds to the total density so we must say no. Then, it will require the name of the density we decided to use. Here, we put the name of the NOCV contribution that we are interested that was listed when we select option `9 - List all available densities` in the main menu.

It is also possible to select the format of the file we are going to generate by using `5 - Select output file format` in the main menu as well as the grid used in option `4 - Enter number of grid intervals`. Finally, the plot file is generated by using option `11 - Generate the plot`. Fig. 5.68 shows the deformation density contributions of the first four NOCVs.

Caution

Do not confuse `base_name.gbw` with `base_name.NOCV.gbw` when wanting to plot the deformation densities. The former contains the deformation density associated to the NOCVs while the later contains the NOCVs orbitals.

Fig. 5.68: Deformation density associated to the first four NOCVs for the Li^+-NH_3

5.37.4 Dealing with fragments with ground state degeneracy

In systems where the electronic configuration for the fragments ground state is degenerate one might be required to take additional precautions to obtain accurate results for the EDA-NOCV decomposition. When dealing with degenerate fragments, the partially occupied molecular orbitals of the promolecule and molecule might not have the same orientation. This is reflected by obtaining integer values for the eigenvalues of the NOCV decomposition and having a higher orbital energy. This is caused by having an inappropiate orbital occupation when creating the promolecule molecular orbitals from the fragments molecular orbitals due to the degeneracy of the ground state electron configuration of the fragments.

One example of this is the C-O system when considering the C and O fragments in their triplet ground state. In this case, the C atom has an electron configuration $2s^2 2p^2$ while the O atom has a configuration $2s^2 2p^4$. The 2p orbitals in both fragments are degenerate. When they come together to make a bond, the molecular fragments are combined as shown in Fig. 5.69. Thus, the orbitals containing unpaired electrons in one fragment should be consistent with the ones of the other fragment. This is done by ensuring that the partially occupied orbitals in both fragments are in the same orientation.

Fig. 5.69: Molecular orbitals of CO molecule

So in case of C-O system, we must ensure that the two unpaired electrons of C atom are in the corresponding $2p_y^1$ and $2p_z^1$ orbitals such as they can combine with the two corresponding unpaired electrons of the $2p_y^1$ and $2p_z^1$ orbitals of the O atom. In this case, it would correspond to exchange the first partially occupied $2p_x^1$ with the unoccupied orbital $2p_z^0$ of the C atoms. In order to do this we need to first run a single-point calculation for the fragment that needs to be modified. Then, we can use the orbitals of that calculation as input for the C fragment calculation within the EDA-NOCV analysis by specifying the `MOREAD` keyword in the method file for the corresponding fragment and then passing the `.gbw` file by using the `%moinp` word. Then we need to specify the molecular orbitals of the fragment we want to exchange. This can be done by using the `Rotate` feature in the `%scf` block.

```
%scf
  Rotate
   { MO1, MO2, Angle }
   { MO1, MO2, Angle, Operator1, Operator2 }
   { MO1, MO2} # Shortcut to swap MO1 and MO2. Angle=90 degrees.
  end
end
```

Where `MO1` and `MO2` refer to the indices of the molecular orbitals we want to exchange. For more detail see Section *Changing the Order of Initial Guess MOs and Breaking the Initial Guess Symmetry*.

Note

The indices for molecular orbitals starts at zero.

The next input corresponds to the calculation of the C-O system from fragment in their triplet ground state. The keyword `FRAG1_FS` is used to flip the spin of the unpaired electrons of the C atom.

```
!BP86 TZVPP  EDA

%EDA
  FRAG1_METHODFILE "frag1.method"
  FRAG2_METHODFILE "BP86 TZVPP"
  FRAG1_M 3
  FRAG1_FS TRUE
  FRAG2_M 3
end

*xyz 0 1
  C(1)           0.00000000000000      0.00000000000000      0.0
  O(2)           0.00000000000000      0.00000000000000      1.136
*
```

In order to have the right occupancies, it is needed to exchange the molecular orbitals of the C fragment. The next input corresponds to the method file for the C fragment. By using the keywords `MOREAD` in the simple input line and `%moinp` the `.gbw` file is passed that contains the molecular orbitals of a previous single point calculation for the C atom. In the `Rotate` subsection, the $2p_x^1$ and $2p_z^0$ orbitals are exchanged, which corresponds to the indeces 2 and 4. So that the p orbital of carbon goes from $2p_x^1 2p_y^1 2p_z^0$ to $2p_x^0 2p_y^1 2p_z^1$ that is aligned with the one of oxygen $2p^2x 2p_y^1 2p_z^1$.

```
BP86 TZVPP MOREAD

%moinp "carbon.gbw"

%scf
  Rotate
  {2,4}
  end
end
```

This ensures consistency between the promolecule and molecule orbitals and minimizes the orbital energy.

5.37.5 EDA-NOCV and Double-Hybrid Functionals

One of the new features in ORCA is the option to use double-hybrid functionals for the EDA-NOCV analysis. These functionals introduce MP2 correlation energy contribution to the energy obtained from the DFT calculation. When using these functionals a new term labeled `MP2 correlation` is added in the energy decomposition analysis. This term corresponds to the difference in the MP2 correlation components between the molecule and its fragments.

When double-hybrid functionals are used it is important to specify the type of density that must be computed: either `relaxed` or `unrelaxed`, and must be specified for the molecule and the fragments by using the `%mp2` block as shown here:

```
!B2PLYP def2-TZVP def2-TZVP/C EDA
%mp2 Density relaxed end
```

The NOCV decomposition is carried out with the density from a DFT calculations. It is possible to also obtain and plot the density associated with the MP2 correlation to the deformation density matrix. This density contribution is defined as $DP_{mp2_contribution} = \Delta P_{mp2_level} - \Delta P_{DFT_level}$. This density is saved as `base_name.DP0.mp2` in case of restricted calculations or `base_name.DP0.mp2` for spin-up contribution and `base_name.DP1.mp2` for spin-down contribution in the case of unrestricted calculations.

5.37.6 List of Related Keywords

Table 5.28: `%eda` block keywords for the EDA/ETS-NOCV decomposition analysis.

Keyword	Options	Description
`FRAG1`	`"<keywords>"`	Method for fragment 1, given as simple input keywords
`FRAG2`	`"<keywords>"`	Method for fragment 2, given as simple input keywords
`FRAG1_METHODFILE`	`"<filename>"`	File that contains the method (simple and/or block) input for fragment 1
`FRAG2_METHODFILE`	`"<filename>"`	File that contains the method (simple and/or block) input for fragment 2
`FRAG1_C`	`0`	Charge of molecular fragment 1
`FRAG2_C`	`0`	Charge of molecular fragment 2
`FRAG1_M`	`1`	Multiplicity of molecular fragment 1
`FRAG2_M`	`1`	Multiplicity of molecular fragment 2
`FRAG1_SF`	`false`	If true, flip the spin of the uncoupled electrons in fragment 1
`FRAG2_SF`	`false`	If true, flip the spin of the uncoupled electrons in fragment 2

5.38 Local Energy Decomposition

The DLPNO-CCSD(T) method provides very accurate relative energies and allows to successfully predict many chemical phenomena. In order to facilitate the interpretation of coupled cluster results, we have developed the Local Energy Decomposition (LED) analysis [773, 774, 775], which permits to divide the total DLPNO-CCSD(T) energy (including the reference energy) into physically meaningful contributions. A practical guide to the LED scheme is reported in Section Section 5.38.1 and Section Section 5.38.2.

Examples of applications of this scheme can be found in Ref. [776, 777, 778, 779, 780, 781].

As a word of caution we emphasize that only the total energy is an observable and its decomposition is, to some extent, arbitrary. Nevertheless, the LED analysis appears to be physically well grounded and logical to us, it is straightforward to apply and comes typically at a negligible computational cost compared to DLPNO-CCSD(T) calculations. Starting from ORCA 4.1, the LED scheme is available for both closed shell and open shell calculations. The code has also been made parallel and more efficient.

The LED scheme makes no assumption about the strength of the intermolecular interaction and hence it remains valid and consistent over the entire potential energy surface. Alternative schemes, such as the popular symmetry adapted perturbation theory, are perturbative in nature and hence are best applied to weakly interacting systems.

The idea of the LED analysis is rather simple. In local correlation methods occupied orbitals are localized and can be readily assigned to pre-defined fragments in the molecule. The same can be done for the correlation energy in terms of pair correlation energies that refer to pairs of occupied orbitals. In this way, both the correlation and the reference energy can be decomposed into intra- and interfragment contributions. The fragmentation is user defined. An arbitrary number of fragments can be defined. In the case that more than 2 fragments are defined, the interfragment interaction is printed for each fragment pair.

A very important feature of the LED scheme is the possibility to distinguish between dispersive and non-dispersive parts of the DLPNO-CCSD(T) correlation energy. In brief, we exploit the fact that each CCSD pair correlation energy contribution can be expressed as a sum of double excitations from pairs of occupied orbitals into the virtual space. As in the DLPNO-CCSD(T) scheme the virtual space is spanned by Pair Natural Orbitals (PNOs) that are essentially local, the entire correlation energy can be decomposed into double excitations types, depending on where occupied and virtual orbitals are localized. For each pair of fragments, the sum of all excitations corresponding to the interaction of instantaneous local dipoles located on different fragments defines the so called "London dispersion" attraction between the two fragments in the LED framework.

For a system of two fragments, one can use as a reference point the geometrically and electronically relaxed fragments that constitute the interacting super-molecule. Relative to this reference state, the binding energy between the fragments can be written as:

$$\begin{aligned}\Delta E =& \Delta E_{geo-prep} \\ &+ \Delta E^{ref.}_{el-prep} + E^{ref.}_{elstat} + E^{ref.}_{exch} \\ &+ \Delta E^{C-CCSD}_{non-dispersion} + E^{C-CCSD}_{dispersion} \\ &+ \Delta E^{C-(T)}_{int}\end{aligned} \tag{5.185}$$

where $\Delta E_{geo-prep}$ is the energy needed to distort the fragments from their equilibrium configuration to the interacting geometry (also called "strain" in other energy decomposition schemes). The $\Delta E^{ref.}_{el-prep}$ term represents the electronic preparation energy and describes how much energy is necessary to bring the fragments into the electronic structure that is optimal for interaction. $E^{ref.}_{exch}$ is the inter-fragment exchange interaction (it always gives a binding contribution in our formalism) and $E^{ref.}_{elstat}$ is the electrostatic energy interaction between the distorted electronic clouds of the fragments. The sum of these terms gives the Hatree-Fock energy in the closed shell case and the energy of the QRO determinant in the open shell case. Finally, the correlation energy is decomposed into dispersive $E^{C-CCSD}_{dispersion}$ and non-dispersive $\Delta E^{C-CCSD}_{non-dispersion}$ contributions plus a triples correction term to the interaction energy $\Delta E^{C-(T)}_{int}$.

The $E^{C-CCSD}_{dispersion}$ term contains the London dispersion contribution from the strong pairs described above plus the interfragment component of the weak pairs, which is essentially dispersive in nature. The $\Delta E^{C-CCSD}_{non-dispersion}$ correlation term serves to correct the contributions to the binding energy approximately included in the reference energy, e.g it counteracts the overpolarization typical of the HF method. It contains the so called charge transfer excitations from the strong pairs $E^{CT(X\rightarrow Y)}_{C-SP} + E^{CT(X\leftarrow Y)}_{C-SP}$, which represent the sum of all double excitation contributions that do not conserve the charge within each fragment. Moreover, the non-dispersive term also includes the electronic preparation from strong ($\Delta E^{C-SP}_{el-prep}$) and weak ($\Delta E^{C-WP}_{el-prep}$) pairs. Finally, $\Delta E^{C-(T)}_{int}$ represents the triples correction contribution to the interaction energy between the fragments. In the LED scheme, this term can be further decomposed into intra- and interfragment components. This can be useful, for example, to estimate the London dispersion contribution from the triples correction term, as suggested in Ref. [777].

"Local Energy Decomposition" (LED) analysis [773, 774, 775] is a tool for obtaining insights into the nature of intermolecular interactions by decomposing the DLPNO-CCSD(T) energy into physically meaningful contributions. For instance, this approach can be used to decompose the DLPNO-CCSD(T) interaction energy between a pair of interacting fragments, as detailed in Section Section 5.38. A useful comparison of this scheme with alternative ways of decomposing interaction energies is reported in Ref. [782, 783, 784].

5.38.1 Closed shell LED

LED decomposes separately the reference (Hartree-Fock) and correlation parts of the DLPNO-CCSD(T) energy. By default, the decomposition of the reference energy makes use of the RI-JK approximation. **An RIJCOSX variant, which is much more efficient and has a much more favorable scaling for large systems, is also available, as detailed in section Section 5.38.7, and in [782].**

> **Note**
>
> For weakly interacting systems, `!TightPNO` settings are typically recommended.

Basic Usage

All that is required to obtain this decomposition in ORCA is to define the *fragments* and specify the `!LED` keyword in the simple input line.

```
dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk LED
```

Example

As an example, the interaction of H_2O with the carbene CH_2 at the PBE0-D3/def2-TZVP-optimized geometry can be analyzed within the LED framework using the following input file:

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO LED

*xyz 0 1
  C(1)      0.16044643459993      0.10093183177121      0.22603351276210
  H(1)      1.04516129053973     -0.06834886965353     -0.41865951747519
  H(1)     -0.12579332868173      1.14737893892114      0.00305818518771
  O(2)     -1.48285705560792     -1.31933824653169      2.29891474420047
  H(2)     -0.91417368674145     -0.93085192992263      1.60917234463506
  H(2)     -1.15648922489703     -2.21246650333085      2.42094328175662
*
```

The corresponding output file is reported below. The DLPNO-CCSD(T) energy components are printed out in different parts of the output as follows:

```
E(0)                                        ...   -114.913309038
E(CORR)(corrected)                          ...     -0.350582526
Triples Correction (T)                      ...     -0.006098691
E(CCSD(T))                                  ...   -115.269990255
```

At the beginning of the LED part of the output, information on the fragments and the assignment of localized MOs to fragments are provided.

```
 ============================================================
         LOCAL ENERGY DECOMPOSITION FOR DLPNO-CC METHODS
 ============================================================

Maximum Iterations for the Localization .. 600
Tolerance for the Localization          ..  1.00e-06
Number of fragments = 2
Fragment  1:  0 1 2
Fragment  2:  3 4 5

Populations of internal orbitals onto Fragments:
   0:      1.000    0.000  assigned to fragment   1
```

(continues on next page)

(continued from previous page)

```
   1:      0.000    1.000  assigned to fragment   2
   2:      1.022    0.008  assigned to fragment   1
   3:      0.001    0.999  assigned to fragment   2
   4:      0.001    0.999  assigned to fragment   2
   5:      1.018    0.000  assigned to fragment   1
   6:      1.019    0.000  assigned to fragment   1
   7:      0.006    1.013  assigned to fragment   2
   8:      0.000    1.016  assigned to fragment   2
```

The decomposition of the Hatree-Fock energy into intra- and inter-fragment contributions follows. It is based on the localization of the occupied orbitals.

```
----------------------------------------
REFERENCE ENERGY E(0) DECOMPOSITION (Eh)
----------------------------------------

Nuclear repulsion     =        28.952049689006
One electron energy   =      -214.430545074583 (T=      114.825132245389, V=     -
↪329.255677319972)
Two electron energy   =        70.565186347093 (J=       71.658914661909, K=      ␣
↪-1.093728314816)
                        -----------------------
Total energy          =      -114.913309038483
Consistency check     =      -114.913309038483 (sum of intra- and inter-fragment␣
↪energies)

Kinetic energy        =       114.825132245389
Potential energy      =      -229.738441283873
                        -----------------------
Virial ratio          =         2.000767922417

-------------------------------------------
INTRA-FRAGMENT REF. ENERGY FOR FRAGMENT   1
-------------------------------------------

Nuclear repulsion     =         6.037208782874
One electron energy   =       -63.553431032444 (T=       38.870491681225, V=     -
↪102.423922713669)
Two electron energy   =        18.675214766985 (J=       18.935443192480, K=      ␣
↪-0.260228425495)
                        -----------------------
Total energy          =       -38.841007482585

Kinetic energy        =        38.870491681225
Potential energy      =       -77.711499163811
                        -----------------------
Virial ratio          =         1.999241476056


-------------------------------------------
INTRA-FRAGMENT REF. ENERGY FOR FRAGMENT   2
-------------------------------------------

Nuclear repulsion     =         9.103529882464
One electron energy   =      -123.025684625357 (T=       75.954640564164, V=     -
↪198.980325189521)
Two electron energy   =        37.916781954190 (J=       38.739989208810, K=      ␣
↪-0.823207254620)
                        -----------------------
Total energy          =       -76.005372788703
```

(continues on next page)

(continued from previous page)

```
Kinetic energy        =        75.954640564164
Potential energy      =      -151.960013352867
                        ----------------------
Virial ratio          =         2.000667927913


-----------------------------------------------------
INTER-FRAGMENT REF. ENERGY FOR FRAGMENTs   2 AND   1
-----------------------------------------------------

Nuclear repulsion     =        13.811311023669
Nuclear attraction    =       -27.851429416782
Coulomb repulsion     =        13.983482260618
                        ----------------------
Sum of electrostatics =        -0.056636132494 (  -35.540 kcal/mol)

Two electron exchange =        -0.010292634701 (   -6.459 kcal/mol)
                        ----------------------
Total REF. interaction =       -0.066928767195 (  -41.998 kcal/mol)



Sum of INTRA-fragment REF. energies  =      -114.846380271288
Sum of INTER-fragment REF. energies  =        -0.066928767195
                                       ---------------------
Total REF. energy                    =      -114.913309038483
```

Afterwards, a first decomposition of the correlation energy is carried out. The different energy contributions to the correlation energy (strong pairs, weak pairs and triples correction) are decomposed into intra- and inter-fragment contributions. This decomposition is carried out based on the localization of the occupied orbitals.

```
--------------------------------
CORRELATION ENERGY DECOMPOSITION
--------------------------------


-----------------------------------------------------
INTER- vs INTRA-FRAGMENT CORRELATION ENERGIES (Eh)
-----------------------------------------------------

                                 Fragment   1            Fragment   2
                        -----------------------  -----------------------
Intra strong pairs              -0.136594658271        -0.209970193798  sum=      ␣
↪-0.346564852069
Intra triples                   -0.002692277706        -0.002842791265  sum=      ␣
↪-0.005535068971
Intra weak pairs                -0.000001573694        -0.000002311734  sum=      ␣
↪-0.000003885429
Singles contribution             0.000000000611         0.000000001058  sum=      ␣
↪ 0.000000001669
                        -----------------------  -----------------------
                                -0.139288509061        -0.212815295738  sum=      ␣
↪-0.352103804799


Interaction correlation for Fragments   2 and   1:
-----------------------------------------------------
Inter strong pairs               -0.003998018810 (   -2.509 kcal/mol)
Inter triples                    -0.000563621667 (   -0.354 kcal/mol)
Inter weak pairs                 -0.000015771468 (   -0.010 kcal/mol)
                         ----------------------
```

(continues on next page)

(continued from previous page)

```
Total interaction                -0.004577411945 (   -2.872 kcal/mol)



Sum of INTRA-fragment correlation energies  =        -0.352103804799
Sum of INTER-fragment correlation energies  =        -0.004577411945
                                              ---------------------
Total correlation energy                    =        -0.356681216744
```

Afterwards, a summary with the decomposition of the total energy (reference energy + correlation) into intra- and inter-fragment contributions is printed.

```
-------------------------------------------
INTER- vs INTRA-FRAGMENT TOTAL ENERGIES (Eh)
-------------------------------------------

                                 Fragment   1            Fragment   2
                        ----------------------  ----------------------
Intra REF. energy              -38.841007482585       -76.005372788703  sum=     -
↪114.846380271288
Intra Correlation energy        -0.139288509061        -0.212815295738  sum=      ␣
↪-0.352103804799
                        ----------------------  ----------------------
                               -38.980295991646       -76.218188084441  sum=     -
↪115.198484076087


Interaction of Fragments   2 and   1:
-------------------------------------
Interfragment reference          -0.066928767195 (  -41.998 kcal/mol)
Interfragment correlation        -0.004577411945 (   -2.872 kcal/mol)
                          ----------------------
Total interaction                -0.071506179140 (  -44.871 kcal/mol)


Sum of INTRA-fragment total energies        =      -115.198484076087
Sum of INTER-fragment total energies        =        -0.071506179140
                                              ---------------------
Total energy                                =      -115.269990255228
```

Hence, the decomposition reported above allows one to decompose all the components of the DLPNO-CCSD(T) energy into intrafragment and interfragment contributions simply based on the localization of the occupied orbitals. In order to convert the intra-fragment energy components into contributions to the binding energy, single point energy calculations must be carried out on the isolated monomers, frozen in the geometry they have in the adduct, and the corresponding terms must be subtracted. Note that one can also include the geometrical deformation energy (also called "strain") by simply computing the energy of the geometrically relaxed fragments (see Section Section 5.38 for further information).

For the DLPNO-CCSD strong pairs, which typically dominate the correlation energy, a more sophisticated decomposition, based on the localization of both occupied orbitals and PNOs, is also carried out and printed. Accordingly, the correlation energy from the strong pairs is divided into intra-fragment, dispersion and charge transfer components. Note that, due to the charge transfer excitations, the resulting intra-fragment contributions (shown below) differ from the ones obtained above.

```
---------------------------------------------
DECOMPOSITION OF CCSD STRONG PAIRS INTO
    DOUBLE EXCITATION TYPES (Eh)
---------------------------------------------

Foster-Boys localization is used for localizing PNOs
```

(continues on next page)

(continued from previous page)

```
Intra fragment contributions:
INTRA Fragment 1                        -0.132849855
INTRA Fragment 2                        -0.209493798

Charge transfer contributions:
Charge Transfer 1 to 2                  -0.005725404
Charge Transfer 2 to 1                  -0.000899609

Dispersion contributions:
Dispersion  2,1                         -0.001594204

Singles contributions:
Singles energy                           0.000000002
```

More detailed information into the terms reported above can be found in Section Section 5.38 and in Ref. [773]. All the individual double excitations contributions constituting the terms reported above can be printed by specifying "printlevel 3" in the `%mdci` block. Finally, a summary with the most important terms of the DLPNO-CCSD energy, which are typically discussed in standard applications, is printed.

```
-------------------------------------------------
FINAL SUMMARY DLPNO-CCSD ENERGY DECOMPOSITION (Eh)
-------------------------------------------------

Intrafragment REF. energy:
Intra fragment   1 (REF.)               -38.841007483
Intra fragment   2 (REF.)               -76.005372789

Interaction of fragments  2 and  1:
Electrostatics (REF.)                   -0.056636132
Exchange (REF.)                         -0.010292635
Dispersion (strong pairs)               -0.001594204
Dispersion (weak pairs)                 -0.000015771

Sum of non dispersive correlation terms:
Non dispersion (strong pairs)           -0.348968665
Non dispersion (weak pairs)             -0.000003885
```

Note that the "Non dispersion" terms include all the components of the correlation energy except London dispersion [776, 783]. For the strong pairs, "Non dispersion" includes charge transfer, intrafragment double excitations and singles contributions. For the weak pairs, it corresponds to the intrafragment correlation contribution. In order to convert the non dispersion correlation components into contributions to the binding energy, single point energy calculations must be carried out on the isolated monomers.

5.38.2 Example: LED analysis of intermolecular interactions

The water-carbene example from the previous section will be used to demonstrate how to analyze intermolecular interactions between two fragments using the LED decomposition (note that all energies are given in a.u. if not denoted otherwise). As often done in practical applications, we will be using geometries optimized at the DFT (PBE0-D3/def2-TZVP) level of theory on which DLPNO-CCSD(T) (cc-pVDZ,TightPNO) single point energies are computed. Note that in practice, basis sets of triple-zeta quality or larger are recommended. In the first step, the geometries of the dimer, H_2O and CH_2 are optimized and DLPNO-CCSD(T) energies are computed to yield E^{opt}_{dimer} and $E^{opt}_{monomers}$. The input examples for the single-point DLPNO-CCSD(T) energies of the monomers at their optimized geometries and the necessary energies from the output files of these runs are as follows:

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO

# H2O optimized at the PBE0-D3/def2-TZVP level
```

(continues on next page)

(continued from previous page)

```
*xyz 0 1
  O     -1.47291471015599     -1.29006364761118      2.29452038079177
  H     -0.88264582939506     -0.99404999457575      1.59835337186103
  H     -1.22136730983407     -2.20010680974562      2.46533021449572
*
```

```
E(0)                                        ...    -76.026656692
E(CORR)(corrected)                          ...     -0.211428886
Triples Correction (T)                      ...     -0.002932804
E(CCSD(T))                                  ...    -76.241018382
```

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO

# CH2 optimized at the PBE0-D3/def2-TZVP level

*xyz 0 1
  C      0.16044643459993      0.10093183177121      0.22603351276210
  H      1.04516129053973     -0.06834886965353     -0.41865951747519
  H     -0.12579332868173      1.14737893892114      0.00305818518771
*
```

```
E(0)                                        ...    -38.881042677
E(CORR)(corrected)                          ...     -0.138447953
Triples Correction (T)                      ...     -0.002873032
E(CCSD(T))                                  ...    -39.022363662
```

Single-point DLPNO-CCSD(T) energies of the monomers at their in-adduct geometries are also necessary. The corresponding inputs and the necessary output parts for these calculations are as follows:

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO

# H2O at the CH2-H2O geometry optimized at the PBE0-D3/def2-TZVP level

*xyz 0 1
  O     -1.48285705560792     -1.31933824653169      2.29891474420047
  H     -0.91417368674145     -0.93085192992263      1.60917234463506
  H     -1.15648922489703     -2.21246650333085      2.42094328175662
*
```

```
E(0)                                        ...    -76.026011663
E(CORR)(corrected)                          ...     -0.211931843
Triples Correction (T)                      ...     -0.002963338
E(CCSD(T))                                  ...    -76.240906844
```

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO

# CH2 at the CH2-H2O geometry optimized at the PBE0-D3/def2-TZVP level

*xyz 0 1
  C      0.16044643459993      0.10093183177121      0.22603351276210
  H      1.04516129053973     -0.06834886965353     -0.41865951747519
  H     -0.12579332868173      1.14737893892114      0.00305818518771
*
```

```
E(0)                                        ...    -38.881085139
E(CORR)(corrected)                          ...     -0.138097323
Triples Correction (T)                      ...     -0.002869022
E(CCSD(T))                                  ...    -39.022051484
```

These energies are summarized in Table 5.29.

Table 5.29: Energies of the H_2O-CH_2 example for illustrating how the different LED contributions are valuated. The superscript ^{opt} denotes energies of optimized structures, ^{fixed} denotes energies of isolated fragments in the dimer structure. In the last column the energy of the dimer is reported.

Energy [a.u.]	H_2O^{opt}	H_2O^{fixed}	CH_2^{opt}	CH_2^{fixed}	H_2O - CH_2
E_{HF}	-76.026656692	-76.026011663	-38.881042677	-38.881085139	-114.913309038
E_c CCSD	-0.211428886	-0.211931843	-0.138447953	-0.138097323	-0.350582526
E_c (T)	-0.002932804	-0.002963338	-0.002873032	-0.002869022	-0.006098691
E_{tot}	-76.241018382	-76.240906844	-39.022363662	-39.022051484	-115.269990255

Note that for this example, we do not include any BSSE correction. For this system we obtain a binding energy of

$$E_{int} = E^{opt}_{dimer} - E^{opt}_{monomers} = -115.269990255 \; - \; (-76.241018382 - 39.022363662) = -0.006608211$$

which is -4.147 kcal/mol.

The basic principles and the details of the LED are discussed in section *Local Energy Decomposition*. The first contribution to the binding energy is the energy penalty for the monomers to distort into the geometry of the dimer (see in Equation (5.185)).

$$\Delta E_{geo-prep} = E^{fixed}_{monomers} - E^{opt}_{monomers}$$

This contribution is computed as the difference of the DLPNO-CCSD(T) energy of the monomers in the structure of the dimer ($E^{fixed}_{monomers}$) and of the relaxed monomers ($E^{opt}_{monomers}$). The following energies are obtained:

$$\Delta E_{geo-prep} = (-76.240906844 + 76.241018382) \; + \; (-39.022051484 + 39.022363662) = 0.000423716$$

which amounts to 0.266 kcal/mol. Typically, the triples correction is evaluated separately:

$$\Delta E^{C-(T)}_{int} = -0.006098691 - (-0.002963338 - 0.002869022) = -0.000266331$$

This contributes -0.167 kcal/mol. The next terms in Equation (5.185) concern the reference energy contributions. The first one is the electronic preparation in the reference, which is evaluated as the difference of the `Intra REF. energy` of the fragments (see previous section) and the reference energy of the separate molecules at the dimer geometry:

$$\Delta E^{ref.}_{el-prep}(H_2O) = -76.005372788703 + 76.026011663 = 0.020638874297$$

$$\Delta E^{ref.}_{el-prep}(CH_2) = -38.841007482585 + 38.881085139 = 0.040077656415$$

which amounts to 0.060716530712 a.u. or 38.100 kcal/mol. The next two contributions stem from the decomposition of the reference inter-fragment contributions $E^{ref.}_{elstat} = -0.056636132$ (-35.540 kcal/mol) and $E^{ref.}_{exch} = -0.010292635$ (-6.459 kcal/mol) and can be found directly in the LED output (`Electrostatics (REF.)` and `Exchange (REF.)`). The correlation energy also contains an electronic preparation contribution, but it is typically included in the correlation contribution $\Delta E^{C-CCSD}_{non-dispersion}$. Adding the non-dispersive strong and weak pairs contributions from the LED output (`Non dispersion (strong pairs)` and `Non dispersion (weak pairs)`) one obtains :

$$-0.348968665 - 0.000003885 = -0.34897255$$

from which we have to subtract the sum of the correlation contributions of the monomers at the dimer geometry

$$\Delta E^{C-CCSD}_{non-dispersion} = -0.34897255 - (-0.211931843 - 0.138097323) = 0.001056616$$

which is 0.663 kcal/mol. The dispersive contribution can be directly found in the LED output (`Dispersion (strong pairs)` and `Dispersion (weak pairs)`) and amounts to $E^{C-CCSD}_{dispersion} = -0.001609975$ which is -1.010 kcal/mol. So all terms that we have evaluated so far are:

$$\Delta E = \Delta E_{geo-prep} + \Delta E^{ref.}_{el-prep} + E^{ref.}_{elstat} + E^{ref.}_{exch} + \Delta E^{C-CCSD}_{non-dispersion} + E^{C-CCSD}_{dispersion} + \Delta E^{C-(T)}_{int}$$

ΔE	$\Delta E_{geo-prep}$	$\Delta E^{ref.}_{el-prep}$	$E^{ref.}_{elstat}$	$E^{ref.}_{exch}$	$\Delta E^{C-CCSD}_{non-disp}$	$E^{C-CCSD}_{disp.}$	$\Delta E^{C-(T)}_{int}$
a.u.	0.000423716	0.06071653071	-0.056636132	-0.010292635	0.001056616	-0.001609975	-0.000266331
kcal/mol	0.266	38.100	-35.540	-6.459	0.663	-1.010	-0.167

which sum to the total binding energies of -0.006608211 a.u. or -4.147 kcal/mol that we have evaluated at the beginning of this section. A detailed discussion of the underlying physics and chemistry can be found in Ref. [774].

5.38.3 Open shell LED

The decomposition of the DLPNO-CCSD(T) energy in the open shell case is carried out similarly to the closed shell case [774]. However, the open shell variant of the LED scheme relies on a different implementation than the closed shell one. A few important differences exist between the two implementations, which are listed below.

- In the closed shell LED the reference energy is typically the HF energy. Hence, the total energy equals the sum of HF and correlation energies. In the open shell variant, the reference energy is the energy of the QRO determinant. Hence, the total energy in this case equals the sum of the energy of the QRO determinant and the correlation energy.
- The singles contribution is typically negligible in the closed shell case due to the Brillouin's theorem. In the open shell variant, this is not necessarily the case. In both cases, the singles contribution is included in the "Non dispersion" part of the strong pairs.
- In the UHF DLPNO-CCSD(T) framework we have $\alpha\alpha$, $\beta\beta$ and $\alpha\beta$ pairs. Hence, in the open shell LED, all correlation terms (e.g. London dispersion) have $\alpha\alpha$, $\beta\beta$ and $\alpha\beta$ contributions. By adding "printlevel 3" in the `%mdci` block one can obtain information on the relative importance of the different spin terms.
- The open shell DLPNO-CCSD(T) algorithm can also be used for computing the energy of closed shell systems by adding the `UHF` keyword in the simple input line of a DLPNO-CCSD(T) calculation.

Basic Usage

Similarly to the closed shell case, define the fragments and include the `!LED` keyword in the simple input line.

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk LED
```

Example

An example of input file is shown below.

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO LED

*xyz 0 3
C(1)      0.32786304018834      0.25137292674595      0.32985672433579
H(1)      0.78308855251826     -0.37244139824620     -0.42204823336026
H(1)     -0.19639272865450      1.19309490346756      0.33713773666060
O(2)     -1.47005964014997     -1.60804001777555      1.84974416203666
H(2)     -0.89305417808014     -1.00736849071095      1.35216686141176
H(2)     -1.02515061661047     -1.73931270222718      2.69260529998224
*
```

The corresponding output is entirely equivalent to the one just discussed for the closed shell case.

5.38.4 COVALED

Local Energy Decomposition is conducted on interaction energies computed using local correlation methods, typically DLPNO-CCSD(T). When a covalent bond is present between two fragments, its inherently delocalized nature makes its assignment to the fragments challenging. To overcome this limitation, the COVALED method has been developed. It represents an extension of the standard LED approach within the DLPNO-CCSD(T) framework, enabling the decomposition of both covalent and non-covalent interaction energies.

Basic Usage

The local energy decomposition of covalent bonds is invoked using the keyword `!LED` in the simple input line, by specifying in the `%mdci` block of the input file the `Covalent` keyword, along with the atoms connected by the covalent bond.

```
! LED

%mdci
  Covalent {0 4}
end
```

Important

If you are using the *Automatic Fragmentation* the Covalent bonds can be assigned automatically by setting `%frag DoInterFragBonds true end`.

Example

As an example, the DLPNO-CCSD(T)/COVALED input file for the ethane dimer is reported. Atoms can be assigned to fragments as explained in *Fragment Specification* section. In this case, four fragments are defined manually, by cutting the covalent bonds between the C atoms in each ethane molecule. Fig. 5.70 illustrates the fragmentation scheme used for this system.

```
! DLPNO-CCSD(T) def2-TZVP def2/J def2-TZVP/C def2/J LED

%mdci Covalent {0 4} {8 12} end

# C4H12 optimized at RI-MP2/aug-cc-pVTZ level of theory

*xyz 0 1
   C(1)   -7.95601053588590      3.15195494946836      0.27839423386884
   H(1)   -8.05179200418556      2.32105035333223      0.97643318840757
   H(1)   -8.76671687926839      3.06907938063429     -0.44469508271450
   H(1)   -8.10439460873783      4.07476211185951      0.83796813304307
   C(2)   -6.59679579710356      3.13992001745864     -0.41129316526153
   H(2)   -6.44433232188420      2.21634557454439     -0.96856734466103
   H(2)   -5.78737090643299      3.22474156709677      0.31305681682191
   H(2)   -6.50317719735328      3.96948543870281     -1.11109265496380
   C(3)   -7.98491788051681     -0.47692316195123     -0.27920144845602
   H(3)   -8.79547255792874     -0.40025515615629      0.44475840345098
   H(3)   -8.14024532507479     -1.38893575993245     -0.85442072275227
   H(3)   -8.07387099754762      0.36622931992467     -0.96327738147245
   C(4)   -6.62637230870360     -0.48881716933962      0.41184856873533
   H(4)   -6.54173070676907     -1.33138502402438      1.09714355762301
   H(4)   -6.46548339397797      0.42323949114744      0.98551040416686
   H(4)   -5.81650657862927     -0.56907193276525     -0.31256550583595
*
```

In the output file, information about the population of shared orbitals are provided.

```
SUMMARY ORBITALS TREATED AS COVALENT:
ORBITAL:   7; POP ATOM   4:  0.500; POP ATOM   0:  0.500
ORBITAL:  14; POP ATOM  12:  0.500; POP ATOM   8:  0.500
```

Afterward, the decomposition of the total energy into intra- and inter-fragment terms is printed. The output structure is the same as that of the standard LED approach (see Section Section 5.38.2), with the main difference lying in the values of the intra- and inter-fragment terms. Unlike the standard LED method, which assigns all the shared electron density to a single fragment, with the COVALED approach the electron density is assigned based on the population analysis of the shared orbitals, thus providing a chemically meaningful decomposition of the total energy.

As the standard LED, the COVALED can be used to decompose intermolecular interaction energy in terms of fragments. As an example, we consider the interaction energy between two ethane molecules in the dimer reported in the example above. The interaction energy is computed as the energy difference between the dimer and the two ethane molecules, keeping their geometry fixed as in the dimer. Its decomposition is shown in the heat maps, where the diagonal elements represent the intra-fragment (electronic preparation) energy terms, while the off-diagonal elements represent the interaction energy between pairs of fragments. Shown below are the heat maps obtained with the standard LED and the COVALED approach.

Fig. 5.70: Fragmentation scheme of the ethane dimer used in the DLPNO-CCSD(T)/COVALED calculation.

(a) a) LED map

(b) b) COVALED map

Fig. 5.71: Heat maps showing the decomposition of the interaction energy between two ethane molecule using the standard LED (a) and COVALED (b) methods.

Note that the total interaction energy remains unchanged, but the way it is distributed among the fragments differs between the standard LED and the COVALED approach.

Keywords

Table 5.30: Simple input keywords for the COVALED.

Keyword	Description
`LED`	Activates LED scheme

Table 5.31: `%mdci` block input keywords for the COVALED.

Keyword	Options	Description
`Coval`	`{<integer> <integer>}`	Defines the indexes of atoms connected through a covalent bond. Is defined automatically if `%frag DoInterFragBonds true end` when using *Automatic Fragmentation*.
`Coval`	`true/false`	If true, the shared electron density is equally partitioned between the two fragments (by default, it is set to true).

5.38.5 Dispersion Interaction Density plot

The Dispersion Interaction Density (DID) plot provides a simple yet powerful tool for the spatial analysis of the London dispersion interaction between a pair of fragments extracted from the LED analysis in the DLPNO-CCSD(T) framework [784]. A similar scheme was developed for the closed shell local MP2 method [785]. The "dispersion energy density", which is necessary for generating the DID plot, can be obtained from a simple LED calculation by adding `DoDIDplot true` in the `%mdci` block.

```
!DLPNO-CCSD(T) ... LED
%mdci DoDIDplot true end
```

These can be converted to a format readable by standard visualization programs, e.g. a cube file, through `orca_plot`. To do that, call `orca_plot` with the command:

```
orca_plot gbwfilename -i
```

and follow the instructions that will appear on your screen, i.e.:

```
1 - Enter type of plot
2 - Enter no of orbital to plot
3 - Enter operator of orbital (0=alpha,1=beta)
4 - Enter number of grid intervals
5 - Select output file format
6 - Plot CIS/TD-DFT difference densities
7 - Plot CIS/TD-DFT transition densities
8 - Set AO(=1) vs MO(=0) to plot
9 - List all available densities

10 - Generate the plot
11 - exit this program
```

Type "1" for selecting the plot type. A few options are possible:

```
1 -   molecular orbitals
2 -   (scf) electron density      ......  (scfp  )  - available
3 -   (scf) spin density          ......  (scfr  )  - available
4 -   natural orbitals
5 -   corresponding orbitals
6 -   atomic orbitals
7 -   mdci electron density       ......  (mdcip )  - NOT available
8 -   mdci spin density           ......  (mdcir )  - NOT available
9 -   OO-RI-MP2 density           ......  (pmp2re)  - NOT available
10 -   OO-RI-MP2 spin density      ......  (pmp2ur)  - NOT available
11 -   MP2 relaxed density         ......  (pmp2re)  - NOT available
12 -   MP2 unrelaxed density       ......  (pmp2ur)  - NOT available
13 -   MP2 relaxed spin density    ......  (rmp2re)  - NOT available
14 -   MP2 unrelaxed spin density  ......  (rmp2ur)  - NOT available
15 -   LED dispersion interaction density  (ded21 )  - available
16 -   Atom pair density
17 -   Shielding Tensors
18 -   Polarisability Tensor
```

Select "LED dispersion interaction density" from the list by typing "15". Afterwards, choose your favorite format and generate the file.

5.38.6 Automatic Fragmentation

Orca can automatically define the *fragments* for the LED analysis. By default, the program will attempt to identify all monomers in the system that are not connected by covalent bonds and assign a fragment to each. However, users can define alternative fragmentation procedures, as explained in the *Automatic Fragmentation* section.

The XYZ coordinates of the fragments are reported in the beginning of the output file. For instance, given the input:

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO LED

*xyz 0 1
C    0.18726407287156      0.08210467619149      0.19811955853244
H    1.07120465088431     -0.00229078749404     -0.46002874025040
H   -0.15524644515923      1.12171178448874      0.04316776563623
O   -1.47509614629583     -1.29358571885374      2.29818864036820
H   -0.87783948760158     -0.98540169212890      1.58987042714267
H   -1.22399221548771     -2.20523304094991      2.47014489963764
*
```

The program will automatically identify the H_2O and the CH_2 fragments.

```
---------------------------------------------
CARTESIAN COORDINATES OF FRAGMENTS (ANGSTROEM)
---------------------------------------------

 FRAGMENT 1
  C     0.187264    0.082105    0.198120
  H     1.071205   -0.002291   -0.460029
  H    -0.155246    1.121712    0.043168

 FRAGMENT 2
  O    -1.475096   -1.293586    2.298189
  H    -0.877839   -0.985402    1.589870
  H    -1.223992   -2.205233    2.470145
```

Note that this procedure works for an arbitrary number of interacting molecules. It is also possible to assign only certain atoms to a fragment and let the program define the other ones:

```
! dlpno-ccsd(t) cc-pvdz cc-pvdz/c cc-pvtz/jk rijk verytightscf TightPNO LED

*xyz 0 1
C(1)     0.18726407287156      0.08210467619149      0.19811955853244
H(1)     1.07120465088431     -0.00229078749404     -0.46002874025040
H(1)    -0.15524644515923      1.12171178448874      0.04316776563623
O   -1.47509614629583     -1.29358571885374      2.29818864036820
H   -0.87783948760158     -0.98540169212890      1.58987042714267
H   -1.22399221548771     -2.20523304094991      2.47014489963764
*
```

Important

If any atom is left unassigned (or explicitly assigned to fragment 0), it will be automatically assigned to a fragment using the *Automatic Fragmentation*. In such cases, the highest manually assigned fragment number must be less than the total number of fragments generated by the combination of manual and automatic procedures. If this condition is not met, ORCA will automatically reorder all fragment numbers in ascending order, starting from 1.

5.38.7 Additional Features, Defaults and List of Keywords

Note

Starting from ORCA 4.2 the default localization scheme for the PNOs has changed from PM (Pipek Mezey) to FB (Foster Boys). This might cause slight numerical differences in the LED terms with respect to that obtained from previous ORCA versions. To obtain results that are fully consistent with previous ORCA versions, PM must be specified (see below).

The following options can be used in accordance with LED.

```
! DLPNO-CCSD(T) cc-pVDZ cc-pVDZ/C cc-pVTZ/JK RIJK TightPNO LED TightSCF

%mdci
  LED 1                # localization method for the PNOs. Choices:
                       # 1 = PipekMezey
                       # 2 = FosterBoys (default starting from ORCA 4.2)
  PrintLevel 3         # Selects large output for LED and prints the
                       # detailed contribution
                       # of each DLPNO-CCSD strong pair
```

(continues on next page)

(continued from previous page)

```
  LocMaxIterLed 600    # Maximum number of localization iterations for PNOs
  LocTolLed     1e-6   # Absolute threshold for the localization procedure for PNOs
  Maxiter 0            # Skips the CCSD iterations and
                       # the decomposition of the correlation energy
  DoLEDHF true         # Decomposes the reference energy in the LED part.
                       # By default, it is set to true.
end
```

Note

Starting from ORCA 4.2 an RIJCOSX implementation of the LED scheme for the decomposition of the reference energy is also available. This is extremely efficient for large systems. For consistency, the RIJCOSX variant of the LED is used only if the underlying SCF treatment is performed using the RIJCOSX approximation, i.e., if RIJCOSX is specified in the simple input line. An example of input follows.

```
!  dlpno-ccsd(t) def2-TZVP def2-TZVP/C def2/j rijcosx verytightscf TightPNO LED

*xyz 0 1
C(1)    0.18726407287156      0.08210467619149      0.19811955853244
H(1)    1.07120465088431     -0.00229078749404     -0.46002874025040
H(1)   -0.15524644515923      1.12171178448874      0.04316776563623
O(2)   -1.47509614629583     -1.29358571885374      2.29818864036820
H(2)   -0.87783948760158     -0.98540169212890      1.58987042714267
H(2)   -1.22399221548771     -2.20523304094991      2.47014489963764
*
```

Finally, here are some tips for advanced users.

- The LED scheme can be used in conjunction with an arbitrary number of fragments.
- The LED scheme can be used to decompose DLPNO-CCSD and DLPNO-CCSD(T) energies. At the moment, it is not possible to use this scheme to decompose DLPNO-MP2 energies directly. However, for closed shell systems, one can obtain DLPNO-MP2 energies from a DLPNO-CCSD calculation by adding a series of keywords in the `%mdci` block: (i) `TScalePairsMP2PreScr 0`; (ii) `UseFullLMP2Guess true`; (iii) `TCutPairs 10` (or any large value). The LED can be used as usual to decompose the resulting energy.
- For a closed shell system of two fragments (say A and B), the LED scheme can be used to further decompose the LED components of the reference HF energy (intrafragment, electrostatics and exchange) into a sum of frozen state and orbital relaxation correction contributions. More information can be found in Ref. [784].
- To obtain the frozen state terms one has to: (i) generate a .gbw file containing the orbitals of both fragments (AB.gbw) using `orca_mergefrag A.gbw B.gbw AB.gbw`, where A.gbw and B.gbw are the orbital files of isolated fragments at the aduct geometry; (ii) run the LED as usual by using `MORead` to read the orbitals in the AB.gbw file in conjunction with `Maxiter 0` in both the `%scf` block (to skip the SCF iterations) and the `%mdci` block (to skip the unnecessary CCSD iterations).

Table 5.32: Simple input keywords for the Local Energy Decomposition.

Keyword	Description
`LED`	Activates the Local Energy Decomposition

Table 5.33: `%mdci` block input keywords for the Local Energy Decomposition.

Keyword	Options	Description
LED	1	PipekMezey used as localization method for the PNOs
	2	FosterBoys used as localization method for the PNOs (default starting from ORCA 4.2)
PrintLevel	3	Selects large output for LED and prints the detailed contribution of each DLPNO-CCSD strong pair
LocMaxIterL	<integer>	Maximum number of localization iterations for PNOs (set to 600 by default)
LocTocLed	<real>	Absolute threshold for the localization procedure for PNOs (set to 1e-6 by default)
MaxIter	0	Skips the CCSD iterations and the decomposition of the correlation energy
DoLEDHF	true/ false	Decomposes the reference energy in the LED part (set to true by default)

5.38.8 Atomic Decomposition Methods

Starting from ORCA 6.1, within the LED framework, it is possible to quantify the interfragment atomic contributions to London dispersion [786](*ADLD*. The decomposition of *DFT-D* London dispersion energies is described in Section 5.40) and exchange energies (*ADEX*):

$$E_{\text{disp}}^{\text{LED}} = \sum_{A} \epsilon_{A}^{\text{disp}} \tag{5.186}$$

$$E_{\text{exch}}^{\text{ref.}} = \sum_{A} \epsilon_{A}^{\text{exch}} \tag{5.187}$$

These atomic decomposition methods are available for both closed- and open-shell DLPNO-based methods (e.g., HFLD, DLPNO-CCSD, DLPNO-CCSD(T)). Different charge partition schemes (Mulliken and Löwdin) are available for the decomposition analysis, with Löwdin set as the default. For each scheme, the analysis reports the corresponding energy for each atom and fragment in the system, with values expressed in atomic units.

Different charge-partitioning schemes can be enabled or disabled by setting the corresponding keyword to `true/false` in the `%mdci` block:

- `AD_Mulliken` for Mulliken-based decomposition
- `AD_Loewdin` for Löwdin-based decomposition

> **Note**
>
> The results are inherently dependent on the chosen molecular fragment definition.

If the contributions from triple excitations are included in the calculation of the LED-based method, ADLD will print the integrated contributions accordingly.

By enabling the `AD_SpinResolved` flag within the `%mdci` block, the calculation will include spin-resolved atomic contributions for open-shell systems. This means that the energy contributions will be explicitly computed for each spin-spin interaction separately: alpha-alpha (↑↑), beta-beta (↓↓), and alpha-beta (↑↓).

> **Note**
>
> Please note that contributions from triple excitations are **not yet** implemented in the spin-resolved decomposition.

Atomic Decomposition of London Dispersion Energy (ADLD(LED))

In systems held together by non-covalent interactions, the dominant correlation contribution to the interaction energy is LD [787]. Therefore, a first estimate of the atomic contribution to the interaction energy, computed using a post-HF scheme, can be obtained by decomposing the correlation contribution to the binding energy into atomic components [788]:

$$E_{\mathrm{C}} = \sum_{A} \epsilon_A \tag{5.188}$$

Since ϵ_A incorporates both the long-range and the short-range dynamic correlation energy, "pure" dispersion energy at the atomic level can be isolated at the *Local Energy Decomposition* level:

$$E_{\mathrm{disp}}^{\mathrm{LED}} = \sum_{A} \epsilon_A^{\mathrm{disp}} \tag{5.189}$$

Basic Usage

The atomic decomposition of dispersion energy is invoked using the `!ADLD` keyword.

```
! DLPNO-CCSD def2-TZVP def2-TZVP/C def2/JK LED ADLD
```

Important

Make sure to also include the `!LED` keyword in the simple input line

Example

As an example, the input file to compute the dispersion interaction between two water molecule at the DLPNO-CCSD(T)/LED level using def2-TZVP basis set along with matching /JK and /C auxiliary basis sets is reported below.

```
! DLPNO-CCSD(T) def2-TZVP def2-TZVP/C def2/JK LED ADLD

%mdci
        AD_Mulliken  true
        AD_Loewdin   true
end

# H2O dimer geometry optimized at B3LYP-D4/def2-TZVP level.

* xyz 0 1
  O(1)       0.00736774224494      0.01160693915772      0.01654049355157
  H(1)       0.96923953706804      0.01163512314627     -0.00786387302697
  H(1)      -0.23645852051665      0.94172445362186     -0.01797690491825
  O(2)      -0.98965374142797     -1.29474095640989      2.41552530001642
  H(2)      -0.69837507632022     -0.90508483715064      1.57804074703869
  H(2)      -1.52582551183659     -2.04783550111168      2.15519678840530
*
```

The corresponding output file is reported below.

```
-----------------------------------------------------------------------
              ADLD - Atomic Decomposition of London Dispersion
-----------------------------------------------------------------------

MULLIKEN-BASED
```

(continues on next page)

(continued from previous page)

```
---------------------------------------

            --------------------------        --------------------------
                        CCSD                            CCSD(T)
            --------------------------        --------------------------

            Correlation      Dispersion       Correlation      Dispersion

1    O       -0.2187971      -0.0006293        -0.2245728      -0.0006459
2    H       -0.0215016      -0.0000253        -0.0220589      -0.0000260
3    H       -0.0214954      -0.0000249        -0.0220526      -0.0000256
4    O       -0.2183144      -0.0005019        -0.2240292      -0.0005151
5    H       -0.0205997      -0.0001578        -0.0211439      -0.0001619
6    H       -0.0225718      -0.0000199        -0.0231578      -0.0000204

Sum  =       -0.5232800      -0.0013591        -0.5370151      -0.0013949

FRAG 1       -0.2617941      -0.0006796        -0.2686843      -0.0006975
FRAG 2       -0.2614859      -0.0006796        -0.2683308      -0.0006974


LOEWDIN-BASED
---------------------------------------

            --------------------------        --------------------------
                        CCSD                            CCSD(T)
            --------------------------        --------------------------

            Correlation      Dispersion       Correlation      Dispersion

1    O       -0.2145399      -0.0006243        -0.2202052      -0.0006408
2    H       -0.0236273      -0.0000279        -0.0242397      -0.0000286
3    H       -0.0236269      -0.0000274        -0.0242394      -0.0000281
4    O       -0.2146356      -0.0004835        -0.2202539      -0.0004961
5    H       -0.0228437      -0.0001750        -0.0234472      -0.0001796
6    H       -0.0240065      -0.0000211        -0.0246297      -0.0000217

Sum  =       -0.5232800      -0.0013591        -0.5370151      -0.0013949

FRAG 1       -0.2617941      -0.0006796        -0.2686843      -0.0006975
FRAG 2       -0.2614859      -0.0006796        -0.2683308      -0.0006974
```

Atomic Decomposition of Exchange (ADEX(LED))

The role of interatomic exchange contributions can be studied by the atomic decomposition of exchange (ADEX) within the LED framework.

Basic Usage

The atomic decomposition of exchange energy is invoked using the `!ADEX` keyword in the simple input line.

```
! DLPNO-CCSD def2-TZVP def2-TZVP/C def2/JK LED ADEX
```

> **Note**
>
> ADEX is currently **not** available with the RIJCOSX approximation.

Important

Make sure to also include the `!LED` keyword in the simple input line

Example

As an example, the input file to compute the exchange energy between two water molecules at DLPNO-CCSD/LED level using def2-TZVP basis set along with matching /JK and /C auxiliary basis sets is reported below.

```
! DLPNO-CCSD def2-TZVP def2-TZVP/C def2/JK LED ADEX

%mdci
        AD_Mulliken  true
        AD_Loewdin   true
end

# H2O dimer geometry optimized at B3LYP-D4/def2-TZVP level.

* xyz 0 1
  O(1)       0.00736774224494      0.01160693915772      0.01654049355157
  H(1)       0.96923953706804      0.01163512314627     -0.00786387302697
  H(1)      -0.23645852051665      0.94172445362186     -0.01797690491825
  O(2)      -0.98965374142797     -1.29474095640989      2.41552530001642
  H(2)      -0.69837507632022     -0.90508483715064      1.57804074703869
  H(2)      -1.52582551183659     -2.04783550111168      2.15519678840530
*
```

The corresponding output file is reported below.

```
-----------------------------------------------------------------------
                 ADEX - Atomic Decomposition of Exchange
-----------------------------------------------------------------------

MULLIKEN-BASED
---------------------------------------

              Exchange

1    O       -0.0028909
2    H       -0.0000526
3    H       -0.0000536
4    O       -0.0022290
5    H       -0.0007038
6    H       -0.0000643

Sum  =       -0.0059942

FRAG 1       -0.0029971
FRAG 2       -0.0029971


LOEWDIN-BASED
---------------------------------------

              Exchange

1    O       -0.0028804
2    H       -0.0000578
3    H       -0.0000589
4    O       -0.0021483
```

(continues on next page)

(continued from previous page)

```
5    H       -0.0007804
6    H       -0.0000684

Sum  =       -0.0059942

FRAG 1       -0.0029971
FRAG 2       -0.0029971
```

Keywords

Table 5.34: Simple input keywords for the atomic decomposition methods.

Keyword	Description
ADEX	Activates atomic decomposition of exchange energy
ADLD	Activates atomic decomposition of dispersion energy

Table 5.35: %mdci block input keywords for the atomic decomposition methods.

Keyword	Options	Description
AD_Mulliken	true/false	Activate Mulliken-based decomposition
AD_Loewdin	true/false	Activate Löwdin-based decomposition (default)
AD_SpinResolved	true/false	Activate spin-resolved decomposition (for open-shell)

5.38.9 Fragment-Pairwise Local Energy Decomposition

Applications of the LED scheme to large multi-fragment systems, such as molecular clusters [789], biomolecular assemblies [790], protein–ligand complexes [781], and molecular crystals [791, 792], have demonstrated its versatility and broad applicability. These studies highlight LED's potential to advance many areas of chemical research, including materials science, catalysis, and drug discovery [793, 794].

For simplicity, the previous sections focused on LED analysis of DLPNO-CCSD(T) interaction energies in two-fragment systems. In practice, the ORCA input/output structure for multi-fragment systems is similar and applies to HFLD as well. However, multi-fragment LED analyses often involve a significantly greater number of intra-fragment and inter-fragment LED terms. Consequently, these analyses require additional tools and techniques to collect and process the extensive data from ORCA outputs.

Among the most effective tools for this purpose is the external LED Analysis Wizard (LEDAW) program package, which automates the extraction and visualization of LED interaction energy matrices and heat maps from ORCA outputs. These matrices/maps tabulate and present all interaction terms, facilitating intuitive interpretation.

As an example, let us consider the inter-strand (K–L) interaction of a DNA duplex composed of three nucleobases per strand, with the water environment treated implicitly. The DNA ladder of the molecular system and the LED interaction energy heat map, which includes the distrubuted CPCM dielectric term as described in Ref. [794], are shown in Fig. 5.72a.

Fig. 5.72: (a) Standard LED and (b) fp-LED heat maps for the BSSE-corrected total inter-strand (K-L) interaction energy of a three-nucleobase-long DNA duplex (backbones omitted) computed at the DLPNO-CCSD(T1)/NormalPNO*/def2-TZVP(-f) level, treating the water environment implicitly with the CPCM model. Each off-diagonal element includes electrostatic, exchange, dispersion, and nondispersive correlation, and CPCM dielectric terms, which are not provided here for simplicity. In the heat maps, red represents attractive interactions, whereas blue represents repulsive interactions. Figure taken from the LEDAW manual.

In this figure, off-diagonal elements represent inter-fragment interactions, while diagonal elements reflect the change in fragment energies upon duplex formation (electronic preparation energies). The submatrix enclosed by a black box highlights genuine interactions between strands K and L. Green-boxed off-diagonal elements are inductive in nature, arising from the perturbation of fragment-pairwise interaction energies upon duplex formation for fragments located at the same strand.

Each diagonal element reflects the cumulative perturbation of a fragment's energy by all other fragments. The fragment-pairwise LED (fp-LED) scheme, introduced in Ref. [794], decomposes this cumulative contribution into individual fragment–pairwise terms, thus eliminating diagonal terms. Each of the resulting pairwise electronic preparation terms represents the sum of mutual perturbations on each fragment's energy, specifically due to the presence of the other.

As seen by comparing panels (a) and (b) of Fig. 5.72, the standard LED heat map contains very large positive and very large negative values. In contrast, when diagonal elements are redistributed into pairwise terms, the resulting fp-LED terms directly reflect the interaction strengths between fragment pairs. Nevertheless, both standard LED and fp-LED lead to consistent conclusions in trend analyses.

The fp-LED terms are more intuitive and comparable to those of isolated dimers, yet they inherently include many-body effects, also known as cooperativity. To analyze such effects, LED can be performed on all isolated fragment pairs across different strands (two-body analysis). Since two-body terms exclude many-body effects, subtracting the two-body LED terms from those of the entire system (N-body analysis) reveals the cooperative contribution to each interaction term. In Fig. 5.72, the diagonal elements of the submatrix enclosed by the black box represent the strongest interactions in this DNA example, highlighting that base-pairing (hydrogen bonding between nucleobases) is the primary contributor to duplex formation. In particular, the G-C pair is stronger than the A-T pair. Although much smaller in magnitude, the stacking interactions (shown by the elements just above and below this diagonal) are also noticeable.

The external LEDAW software supports both the automated preparation of ORCA input files for LED calculations and the post-processing of ORCA/LED output files. It automates all types of LED analyses, including N-body, two-body, and cooperativity analyses with or without CPS and CBS extrapolations, within the frameworks of standard LED and fp-LED schemes. LEDAW calculates the interaction energy matrix for each LED component and generates the corresponding heat map, similar to those shown in Fig. 5.72. Additionally, LEDAW includes a distribution scheme for the CPCM dielectric contribution to pairwise terms and another scheme to estimate the dispersion component

of triple excitations. Its user-friendly GUI, equipped with built-in info buttons and help messages, provides clear guidance at each step.

Let us now demonstrate multi-fragment ORCA/LED input files for the DNA duplex formation example (see Fig. 5.72) at both the DLPNO-CCSD(T) and HFLD levels. Outputs for this example, along with additional multi-fragment interaction problems, are available in the LEDAW repository.

Multi-fragment DLPNO-CCSD(T)/LED Inputs

Below is a sample DLPNO-CCSD(T)/LED input for the full six-fragment DNA system (supersystem), using `NormalPNO*` settings, i.e., `NormalPNO` settings with the `TCutPairs` threshold adopted from the `TightPNO` settings:

```
! DLPNO-CCSD(T) def2-TZVP(-f) RIJCOSX DefGrid3 def2/J def2-TZVP/C
! VeryTightSCF CPCM(water) NormalPNO LED

%mdci TCutPairs 1e-5 end

*xyz 0 1
 H(1)      10.634700344584      13.691281144512     -11.875553166247
 N(1)       9.980106156592      13.839587738160     -11.071433575066
 C(1)      10.442864004832      13.875896090101      -9.798268811879
 H(1)      11.504832150180      13.715386729876      -9.677066287960
 C(1)       9.606316071622      14.082566153675      -8.757444583846
 H(1)       9.969796490105      14.122858111485      -7.746934660848
 C(1)       8.213760646535      14.241454159912      -9.044751776248
 N(1)       7.327081712146      14.401901044136      -8.074002131001
 H(1)       7.652431838508      14.636728247003      -7.150492447526
 H(1)       6.356598650077      14.618122743997      -8.319845047378
 N(1)       7.770953959427      14.126729133636     -10.296510374266
 C(1)       8.599854136225      13.955873478910     -11.327889587706
 O(1)       8.191469311313      13.869760857544     -12.487179766903
 H(2)       7.908971458264      16.330584347190     -14.914226197121
 N(2)       7.881940637725      16.532665928810     -13.880511672013
 C(2)       9.012048768510      16.770206523525     -13.152689461214
 H(2)       9.940538578336      16.685266706211     -13.698351026605
 C(2)       9.004919716563      17.074088156969     -11.839690594854
 C(2)      10.229508549268      17.422595087270     -11.055129656290
 H(2)      11.104663398618      17.455282888334     -11.696506986450
 H(2)      10.077971944522      18.396373650562     -10.593668745583
 H(2)      10.390332193916      16.701336107263     -10.255896978574
 C(2)       7.728677472520      17.111302212168     -11.152199754353
 O(2)       7.591576070140      17.345499551464      -9.956268414904
 N(2)       6.645820288608      16.805693851400     -11.932197008459
 H(2)       5.695108359505      16.933012057655     -11.497671886556
 C(2)       6.643971848457      16.537517940386     -13.265921081701
 O(2)       5.608659448998      16.284373997746     -13.862115698184
 H(3)       4.167079527237      19.199246063134     -16.022543736994
 N(3)       4.720709708947      19.310013304079     -15.150325886002
 C(3)       6.064240770076      19.215142752868     -14.931704323361
 H(3)       6.747843821168      18.967933282822     -15.718663332453
 N(3)       6.412548423676      19.456483831030     -13.703531446170
 C(3)       5.232781972225      19.735449439261     -13.049658288826
 C(3)       4.949492015809      20.145779207010     -11.720076857315
 O(3)       5.745437995832      20.316096696627     -10.796304698950
 N(3)       3.593297648465      20.377473357167     -11.520990626887
 H(3)       3.323052692831      20.746411963262     -10.587381622814
 C(3)       2.622889432685      20.208130250514     -12.467641661931
 N(3)       1.351243114016      20.384866184300     -12.095051028829
 H(3)       1.138815575017      20.797025787658     -11.187898890469
 H(3)       0.665646529482      20.443543657416     -12.828192563467
```

(continues on next page)

(continued from previous page)

```
 N(3)       2.872666242150      19.837350187609     -13.703978842593
 C(3)       4.173814909084      19.635730662889     -13.950649270381
 H(4)       0.607492786974      13.693167256443     -12.498905296936
 N(4)       1.156233913621      13.976335931793     -11.654346631242
 C(4)       0.719515992071      14.493104455789     -10.463418891368
 H(4)      -0.320190226981      14.652465882135     -10.253396768309
 N(4)       1.675522233408      14.756439112108      -9.624659463980
 C(4)       2.833919441760      14.430973751698     -10.293445908760
 C(4)       4.205441626210      14.518429914241      -9.917743216750
 O(4)       4.678969477231      14.902137378802      -8.847036060363
 N(4)       5.046351902043      14.044212601799     -10.915024253057
 H(4)       6.072714102757      14.111198911849     -10.723202969726
 C(4)       4.637211663222      13.564345980913     -12.125004367406
 N(4)       5.571436643365      13.009014240888     -12.919396141151
 H(4)       6.536208806688      13.328415372914     -12.798515557053
 H(4)       5.278456082314      12.891881947995     -13.876688905133
 N(4)       3.372577813131      13.480706123713     -12.484778526663
 C(4)       2.516506103177      13.957107881401     -11.567141462038
 H(5)      -0.427318750124      17.859731737670     -10.029201555848
 N(5)       0.505872319628      18.084321359603      -9.624732567559
 C(5)       0.836583758659      18.604519628671      -8.407322768699
 H(5)       0.092411961428      18.948262192878      -7.715100794658
 N(5)       2.115426925402      18.643562050428      -8.177257646546
 C(5)       2.692395320505      18.137378704062      -9.318466049416
 C(5)       4.023866632235      17.839636438080      -9.678850958220
 N(5)       5.069903086692      17.996703370707      -8.878260648241
 H(5)       4.919055989330      18.448434690564      -7.993811348555
 H(5)       6.012493323398      17.797866055580      -9.225623461520
 N(5)       4.208096816167      17.308977601422     -10.903952927055
 C(5)       3.171511385838      17.075799836042     -11.696070915968
 H(5)       3.416948001374      16.659607513112     -12.669038097596
 N(5)       1.893387444935      17.271543482840     -11.450623229708
 C(5)       1.692338416848      17.804060912579     -10.238211580065
 H(6)       0.347959720865      22.070953898527      -7.231147376111
 N(6)       1.367130517057      22.050245111897      -7.460759445434
 C(6)       2.293054934819      22.474694200979      -6.575605167157
 H(6)       1.905330062598      22.892493241868      -5.657222755113
 C(6)       3.613987194667      22.359676299416      -6.836971827951
 H(6)       4.357477801351      22.686127035235      -6.131652395756
 C(6)       3.986574767236      21.774827021668      -8.089632503708
 N(6)       5.248199676647      21.489136249217      -8.363368075578
 H(6)       5.974584821090      21.754378375091      -7.719014977155
 H(6)       5.487376252505      21.069865859566      -9.271452130257
 N(6)       3.057799105691      21.473978059250      -8.997786542956
 C(6)       1.758387335929      21.635696482051      -8.749410306360
 O(6)       0.886565761830      21.396547045533      -9.589416228698
*
```

Fragments 1-3 belong to strand K and form subsystem 1, while fragments 4-6 belong to strand L and form subsystem 2. If BSSE correction is requested, the subsystem inputs can be prepared by modifying the supersystem input as follows:

- Subsystem 1 input: Prefix fragment labels (4), (5), and (6) with a ":" sign to designate them as ghost atoms.
- Subsystem 2 input: Prefix fragment labels (1), (2), and (3) with a ":" sign to designate them as ghost atoms.

If BSSE correction is not required, the ghost fragments must be removed from the respective subsystem input files. In this case, subsystem 1 input contains only the strand K fragments, while the subsystem 2 input contains only the strand L fragments. Fragment labels in subsystem 2 could also be renumbered as 1–3, rather than keeping the original 4–6 labels. This prevents ORCA from retaining placeholder entries for fragments 1–3, resulting in cleaner and more readable outputs.

Multi-fragment HFLD/LED Inputs

Below is a sample HFLD/LED input for the DNA supersystem (supersystem), using the default `NormalPNO*` settings. Coordinate and fragment label specifications are not provided anymore as they are the same as in the previous subsection.

```
! HFLD def2-TZVP(-f) RIJCOSX DefGrid3 def2/J def2-TZVP/C
! VeryTightSCF CPCM(water)

%mdci DoLedHF True   # HFLD default: False
      HFFragInter {1 2} {1 3} {2 3} {4 5} {4 6} {5 6}
      end
```

In the HFLD method, intra-subsystem correlation is neglected by design, meaning each subsystem is treated at the reference level. However, by default, the HFLD implementation in ORCA treats each fragment as a separate subsystem. To explicitly specify fragment pairs to be excluded from correlation, the `HFFragInter` directive must be used. In the above example, this directive sets the correlation energy among strand K fragments (1-3) and among strand L fragments (4-6) to zero. Note that intra-fragment correlations do not need to be specified explicitly, as they are automatically excluded when the HFLD keyword is used.

If a subsystem consists of a single fragment, only a reference energy calculation (RHF or QRO) is required to compute the RHF/HFLD or QRO/HFLD interaction energy. For subsystems containing multiple fragment, an RHF/LED or QRO/LED calculation is needed to decompose the interaction energy into pairwise contributions. If both DLPNO-CCSD(T) and HFLD calculations are to be performed, the LED data for the refence energy are already included in the DLPNO-CCSD(T) subsystem outputs. Otherwise, since LED analysis cannot be invoked at the reference level alone in the current ORCA version, a low-cost DLPNO-CCSD or HFLD correlation calculation can be used in the subsystem inputs to efficiently obtain the necessary LED data, as illustrated below:

```
! HFLD def2-TZVP(-f) RIJCOSX DefGrid3 def2/J def2-TZVP/C
! VeryTightSCF CPCM(water) LoosePNO

%mdci DoLedHF True   # HFLD default: False
      maxiter 0
      end
```

In this example, the `maxiter 0` directive in the `%mdci` block is used to bypass unnecessary coupled-cluster iterations. When configuring a low-cost correlation calculation to obtain reference LED data, do not truncate the correlation space too aggressively since at least one electron pair must remain for ORCA to perform LED analysis. Here, LoosePNO settings are employed. However, if this excludes all electron pairs, slightly tighter PNO settings may be required. For large systems, selectively excluding certain interfragment correlations via the `HFFragInter` directive can further reduce the computational cost. However, avoid specifying all interfragment pairs since this will exclude all electron pairs, which disables LED analysis entirely.

5.39 The Hartree-Fock plus London Dispersion (HFLD) Method

The efficient and accurate HFLD method[795] can be used for the quantification and analysis of noncovalent interactions between a pair of user-defined fragments. Starting from ORCA 5.0, an open shell variant of the HFLD method is also available.[796]

The leading idea here is to solve the DLPNO coupled cluster equations while neglecting intramonomer correlation. The LED scheme is then used to extract the London dispersion (LD) energy from the intermolecular part of the correlation. Finally, the resulting LD energy is used to correct interaction energies computed at the HF level. Hence, HFLD can be considered as a dispersion-corrected HF approach in which the dispersion interaction between the fragments is added at the DLPNO-CC level. As such, it is particularly accurate for the quantification of noncovalent interactions such as those found in H-bonded systems, pre-reactive intermediates (e.g., Frustrated Lewis Pairs), dispersion and electrostatically bound systems. Larger errors are in principle expected for transition metal complexes, as it is the case for any dispersion corrected Hartree-Fock scheme.

The efficiency of the approach allows the study of noncovalent interactions in systems with several hundreds of atoms.

Some of the most important aspects of the method are summarized below:

- **Accuracy and Recommended Settings**
  For noncovalent interactions, HFLD typically provides an accuracy comparable to that of the DLPNO-CCSD(T) method if default PNO settings are used. For the HFLD scheme, these are defined as TCutPNO = 3.3e-7 and TCutPairs 1e-5. If used in conjuction with a def2-TZVP(-f) basis set, these settings are typically denoted as `HFLD*` and are recommended for standard applications on large systems [796]. For example, `HFLD*` settings were used in Ref. [790] to elucidate the complex pattern of interactions responsible for the stability of the DNA duplex. If great accuracy is required, it is recommended to use TightPNO settings in conjuction with TCutPNO 1e-8 and two-point basis set extrapolation (aug-cc-pVTZ/aug-cc-pVQZ) to approach the CBS limit. These settings are typically denoted as the **gold** HFLD settings [796].

- **Reference determinant in the Open shell HFLD scheme**
  In the open shell case, HFLD relies on a restricted reference determinant for the calculation of the LD energy. If the QRO determinant is used as reference, the reference interaction energy can in principle be computed at the UHF or QRO levels. This leads to two different schemes, namely the QRO/HFLD and UHF/HFLD. Alternatively, the restricted open-shell HF (ROHF) determinant can be used as reference in HFLD calculations, which leads to the ROHF/HFLD approach. The energy value reported as `FINAL SINGLE POINT ENERGY` in the output corresponds to the UHF/HFLD scheme by default, which is typically slightly more accurate. See Ref. [796] for details.

- **Efficency**
  The calculation of the dispersion correction typically requires the same time as an HF calculation. This is true for small as well as for large systems.

- **Analysis of Intermolecular Interactions**
  The HFLD method can be combined with the Local Energy Decomposition (LED) to study intermolecular interactions in great detail. The LED dispersion energy obtained with HFLD is often very close to that obtained from a full DLPNO-CCSD(T) calculation. Hence, HFLD can be used as a cost-effective alternative to DLPNO-CCSD(T) to study, among other things, the importance of London dispersion in molecular chemistry.

- **Additional considerations**
  (i) One can specify `NormalPNO` or `TightPNO` settings in the simple input line. The corresponding DLPNO thresholds would be in this case fully consistent with those used in the DLPNO-CCSD(T) method. (ii) The dispersion energy in the HFLD approach slightly depends on the choice of the localization scheme used for occupied orbitals and PNOs. Default settings are recommended for all intents and purposes. However, it is important to note that the localization iterations for occupied and virtual orbitals must be fully converged in order to obtain consistent results. To achieve this goal, it might be necessary to increase `LocMaxIter` or `LocMaxIterLed` (see below). However, this is typically necessary only if very large basis sets (e.g. aug-cc-pV5Z) are used. (iii) Importantly, the method benefits from the use of tightly converged SCF solutions. For closed-shell systems, a useful diagnostic in this context is the "Singles energy" term that is printed in the LED part of the output. This term must be smaller than 1e-6 for closed shell species. If this is not the case, one should change the settings used for the SCF iterations. Note also that all the features of the LED scheme (e.g. automatic fragmentation) are also available for the HFLD method.

Note that, as HFLD relies on both the DLPNO-CCSD(T) and LED methods, the options of both schemes can be used in principle in conjunction with HFLD. Some examples are shown below:

```
! HFLD aug-cc-pVDZ aug-cc-pVDZ/C aug-cc-pVTZ/JK RIJK TightSCF

%mdci
  LED 1               # localization method for the PNOs. Choices:
                      # 1 = PipekMezey
                      # 2 = FosterBoys (default, recommeded for the HFLD method)
  PrintLevel 3        # Selects large output for LED and prints the
                      # detailed contribution
                      # of each DLPNO-CCSD strong pair
  LocMaxIterLed 600   # Maximum number of localization iterations for PNOs
  LocMaxIter    300   # Maximum number of localization iterations for
                      # occupied orbitals
  LocTolLed     1e-6  # Absolute threshold for the localization procedure for PNOs
  DoLEDHF true        # Decomposes the reference energy in the LED part.
```

(continues on next page)

(continued from previous page)

```
                      # By default, it is set to false in HFLD for efficiency␣
↪reasons.
  TCutPNO    3.33e-7  # cutoff for PNO occupation numbers.
  TCutPairs  1e-5     # cutoff for estimated pair correlation energies
                      # to be included in the CC treatment
end
```

5.39.1 Basic Usage

The Hartree-Fock plus London Dispersion method is invoked using the `!HFLD` keyword in the simple input line.

```
! HFLD aug-cc-pvdz aug-cc-pvdz/C
```

5.39.2 Example

An input example is reported below.

```
!  HFLD aug-cc-pvdz aug-cc-pvdz/C verytightscf

*xyz 0 1
C(1)    0.18726407287156      0.08210467619149      0.19811955853244
H(1)    1.07120465088431     -0.00229078749404     -0.46002874025040
H(1)   -0.15524644515923      1.12171178448874      0.04316776563623
O(2)   -1.47509614629583     -1.29358571885374      2.29818864036820
H(2)   -0.87783948760158     -0.98540169212890      1.58987042714267
H(2)   -1.22399221548771     -2.20523304094991      2.47014489963764
*
```

In the corresponding output, after the DLPNO-CC iterations and the LED output, the following information is printed:

```
----------------------------   ----------------
Inter-fragment dispersion       -0.001871763
----------------------------   ----------------


-------------------------   --------------------
FINAL SINGLE POINT ENERGY      -114.932878050741
-------------------------   --------------------
```

The total HFLD energy of the adduct is thus -114.932878050741 a.u.. To compute interaction energies, we have to subtract from this value the Hartree-Fock energies of the monomers in the geometry they have in the complex, i.e., -38.884413525377 and -76.040412827089 a.u. for CH_2 and H_2O, respectively. The total interaction energy is thus -0.00805 a.u. or -5.1 kcal/mol (the corresponding DLPNO-CCSD(T)/TightPNO/CBS value is -5.3 kcal/mol [774]). Note that, to obtain binding energies, the geometric preparation should be added to this value. This can be computed using a standard computational method, e.g, DFT or DLPNO-CCSD(T).

5.39.3 Keywords

> **Note**
>
> As HFLD relies on both the DLPNO-CCSD(T) and LED methods, the options of both schemes can be used in principle in conjunction with HFLD.

Table 5.36: Simple input keywords for the Hartree Fock plus London Dispersion.

Keyword	Description
`HFLD`	Activates the Hartree Fock plus London Dispersion method

Table 5.37: `%mdci` block input keywords for the Hartree Fock plus London Dispersion.

Keyword	Options	Description
`LED`	`1`	PipekMezey used as localization method for the PNOs
	`2`	FosterBoys used as localization method for the PNOs (default starting from ORCA 4.2)
`PrintLevel`	`3`	Selects large output for LED and prints the detailed contribution of each DLPNO-CCSD strong pair
`LocMaxIterI`	`<integer>`	Maximum number of localization iterations for PNOs (set to 600 by default)
`LocMaxIter`	`<integer>`	Maximum number of localization iterations for occupied orbitals (set to 128 by default)
`LocTocLed`	`<real>`	Absolute threshold for the localization procedure for PNOs (set to 1e-6 by default)
`DoLEDHF`	`true/ false`	Decomposes the reference energy in the LED part (by default, it is set to false in HFLD for efficency reasons).
`TCutPNO`	`<real>`	Cutoff for PNO occupation numbers (set to 3.33e-7 by default).
`TCutPairs`	`<real>`	Cutoff for estimated pair correlation energies to be included in the CC treatment (set to 1e-5 by default).

5.40 ADLD(D): Atomic Decomposition of DFT-D London Dispersion Energy

Starting from version 6.1, ORCA offers a method to quantify the London dispersion energy (LD) with atomic resolution in conjunction with a wide range of electronic structure techniques, as described in [788] and [786]. A special case of this decomposition scheme is the atomic decomposition of the *DFT-D* London Dispersion energies (e.g. ADLD(D4) for DFT-D4). This scheme is available for D2, D3, and D4. The original scheme for DLPNO-based LED methods is documented in the *ADLD(LED) section*.

5.40.1 Theory

The atomic LD energy for DFT-D is assessed by assigning half of the pairwise contribution to each atom of the pair and iterating over all atoms:

$$E_{\text{disp}}^{\text{DFT-D}}(AB) = \sum_{A>B} \epsilon_{AB}^{\text{DFT-D}} = \frac{1}{2}\sum_{A}\sum_{B\neq A} \epsilon_{AB}^{\text{DFT-D}} = \sum_{A} \epsilon_{A}^{\text{DFT-D}}(AB) \tag{5.190}$$

If three-body effects are included in the calculation, the decomposition includes them as well:

$$E_{\text{disp}}^{\text{DFT-D}}(ABC) = \sum_{A>B>C} \epsilon_{ABC}^{\text{DFT-D}} = \frac{1}{6}\sum_{A}\sum_{B\neq A\neq C} \epsilon_{ABC}^{\text{DFT-D}} = \sum_{A} \epsilon_{A}^{\text{DFT-D}}(ABC) \tag{5.191}$$

Basic Usage

The atomic decomposition of LD energy is invoked using the `!ADLD` keyword in the simple input line.

```
! PBE D3 def2-TZVP ADLD
```

Example

As an example, the atomic dispersion energies at the B3LYP-D4/def2-TZVP level of a water dimer can be analyzed using the following input file:

```
! B3LYP D4 def2-TZVP ADLD

# H2O dimer geometry optimized at the B3LYP-D4/def2-TZVP level.

* xyz 0 1
  O       0.00736774224494      0.01160693915772      0.01654049355157
  H       0.96923953706804      0.01163512314627     -0.00786387302697
  H      -0.23645852051665      0.94172445362186     -0.01797690491825
  O      -0.98965374142797     -1.29474095640989      2.41552530001642
  H      -0.69837507632022     -0.90508483715064      1.57804074703869
  H      -1.52582551183659     -2.04783550111168      2.15519678840530
*
```

The corresponding output file is reported below. The LD energy in atomic units is printed for each atom, along with the total dispersion. The output format remains consistent for the other -D corrections.

```
-------------------------------------------------------------
      ADLD - Atomic Decomposition of London Dispersion
-------------------------------------------------------------

Atom   Type     Dispersion (a.u.)
---------------------------------
 1      O            -0.000523477
 2      H            -0.000128807
 3      H            -0.000128264
 4      O            -0.000477971
 5      H            -0.000163561
 6      H            -0.000120093

Total dispersion     -0.001542174
```

Keywords

Table 5.38: Simple input keywords for the atomic decomposition of London dispersion calculated with DFT-D correction.

Keyword	Description
`ADLD`	Activates atomic decomposition of LD

5.41 Static Ground State DFT (SGS-DFT)

5.41.1 Suitability of SGS-DFT for X-ray Spectroscopy

SGS-DFT reduces the complexity of many-electron systems to a one-electron framework by leveraging the Kohn-Sham equations, focusing on electron density that is used to compute ground-state orbitals. In this concept, excited state problems can be constructed on the basis of orbital energy differences, in strict difference to excited state methodologies (e.g., TD-DFT, ROCIS, RASCI/CASCI/NEVPT2, MRCI/MREOM-CC). This simplicity makes it computationally efficient while still providing reliable results in a number of X-ray spectroscopy experiments, such as X-ray Absorption Spectroscopy (XAS), X-ray Emission Spectroscopy (XES), and Resonant Inelastic X-ray Scattering (RIXS).

SGS-DFT has found significant applications in Valence-to-Core X-ray Emission Spectroscopy (VtC-XES), which probes valence orbitals via core-hole transitions, offering insights into ligand-metal interactions. It complements other one-electron methods like TD-DFT for XAS, as well as wavefunction-based approaches (ROCIS, RASCI/CASCI/NEVPT2, MRCI/MREOM-CC) and Ligand Field-based CI methods, balancing accuracy and computational cost. A collective review about all these methodologies their advantages and their limitationsisin computing X-ray spectroscopy spectra is provided in reference [458].

5.41.2 Summary of Cross Section Equations

In this section we sumarize the general, non-relativistic and releativistic expressions for cross sections in X-ray spectroscopy, including XAS, XES, and RIXS.

Indices and Notations

ρ, ρ': Cartesian components of the dipole moment, summing over x, y, z, i.e., $\rho, \rho' \in x, y, z$.

I: Initial state (ground state in general expressions).

F: Final state in general cross-section and XAS expressions, with summation over F including all possible final states.

V, V': Intermediate states in RIXS expressions, with energies $\epsilon_V, \epsilon_{V'}$.

i_σ, j_σ: Occupied 1-electron orbitals in non-relativistic treatment, where $\sigma \in \alpha, \beta$ denotes spin.

a_σ: Unoccupied 1-electron orbitals in non-relativistic treatment, with $\sigma \in \alpha, \beta$.

ν_σ, ν'_σ: Intermediate states in non-relativistic RIXS expressions, with $\sigma \in \alpha, \beta$.

ω_{ex}: Excitation energy (incident radiation frequency).

ω_{em}: Emission energy (scattered radiation frequency in XES).

ω_{sc}: Scattering energy (used in RIXS for scattered radiation frequency).

$\Delta\omega$: Energy transfer in RIXS, typically $\Delta\omega = \omega_{ex} - \omega_{sc}$.

ω_{FI}: Transition energy between initial state I and final state F.

ω_{VI}: Transition energy between initial state I and intermediate state V.

$\omega_{a_\sigma i_\sigma}$: Transition energy between unoccupied orbital a_σ and occupied orbital i_σ, defined as $\epsilon_{a_\sigma} - \epsilon_{i_\sigma}$.

$\omega_{j_\sigma i_\sigma}$: Transition energy between occupied orbitals j_σ and i_σ, defined as $\epsilon_{j_\sigma} - \epsilon_{i_\sigma}$.

$\epsilon_{a_\sigma}, \epsilon_{i_\sigma}, \epsilon_{\nu_\sigma}, \epsilon_{\nu'\sigma}$: Orbital energies for unoccupied orbital $a\sigma$, occupied orbital i_σ, and intermediate states ν_σ, ν'_σ.

$\Gamma, \Gamma_{\nu_\sigma}, \Gamma_{\nu'_\sigma}$: Linewidth parameters for lifetime broadening.

Γ_F: Linewidth parameter for final state F.

c: Speed of light (~ 137 in atomic units).

m: Electric dipole transition operator, defined as $m = \sum_i \mathbf{r}_i + \sum_A \mathbf{Z}_A \mathbf{R}_A$, where $\mathbf{r}_i$ are electron coordinates and $\mathbf{Z}_A \mathbf{R}_A$ are nuclear contributions.

General Expressions

- **XAS Cross Section**

The XAS cross section, describing absorption from initial state I to final state F, is:

$$\sigma_{\text{XAS}}(\omega_{ex}) = \frac{4\pi}{3c}\omega_{ex}\sum_{F}\sum_{\rho\in\{x,y,z\}}\frac{|\langle F|m_\rho|I\rangle|^2}{(\omega_{FI}-\omega_{ex})^2+\frac{1}{4}\Gamma^2}$$

or equivalently:

$$\sigma_{\text{XAS}}(\omega_{ex}) = \frac{4\pi}{3c}\omega_{ex}\sum_{F}\sum_{\rho\in\{x,y,z\}}|\langle F|m_\rho|I\rangle|^2 f(\omega_{ex},\omega_{ex},\Gamma)$$

- **XES Cross Section**

The XES cross section, describing emission from an intermediate state to a final state F, is:

$$\sigma_{\text{XES}}(\omega_{em}) = \frac{4\pi}{3c}\omega_{em}\sum_{F}\sum_{\rho\in\{x,y,z\}}\frac{|\langle F|m_\rho|I\rangle|^2}{(\omega_{FI}-\omega_{em})^2+\frac{1}{4}\Gamma^2}$$

- **RIXS Cross Section**

The inelastic scattering cross section for RIXS is:

$$\sigma(\omega_{ex},\omega_{sc}) = \frac{8\pi}{9c^4}\omega_{ex}^3\omega_{sc}\sum_{\rho,\rho'\in\{x,y,z\}}|\alpha_{\rho,\rho'}(\omega_{ex},\omega_{sc})|^2$$

- **Polarizability Tensor**

The polarizability tensor from the Kramers-Heisenberg-Dirac expression is:

$$\alpha_{\rho,\rho'}(\omega_{ex},\omega_{sc}) = \sum_{V}\sum_{F}\frac{\langle F|m_{\rho'}|V\rangle\langle V|m_\rho|I\rangle}{(\omega_{VI}-\omega_{ex}-\frac{i}{2}\Gamma)}+\frac{\langle F|m_\rho|V\rangle\langle V|m_{\rho'}|I\rangle}{(\omega_{VI}+\omega_{sc}-\frac{i}{2}\Gamma)}$$

5.41.3 Orbitals, States, and Energies

The present treatment is based on a single one-electron approach. The orbitals come from a density functional theory (DFT) calculation and satisfy the Kohn-Sham equations

$$F^{\text{KS}}|\rho_{\mathbf{k}}\rangle = \varepsilon_{\mathbf{k}}|\rho_{\mathbf{k}}\rangle$$

where F^{KS} is the Kohn-Sham operator, $|\rho_{\mathbf{k}}\rangle$ is a spin-orbital with spin $\sigma_{\mathbf{k}} = 0, \beta$ and $\varepsilon_{\mathbf{k}}$ is the orbital energy. The Kohn-Sham operator is:

$$F^{\text{KS}} = h + \int\left(\frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\right)d\mathbf{r}' + V^{\text{xc}}[\rho(\mathbf{r})]$$

where h is the one-electron Hamiltonian (kinetic energy plus electron-nuclear potential together with possibly present other external potentials), the second term presents the Coulomb repulsion where

$$\rho(\mathbf{r}) = \rho^\alpha(\mathbf{r})+\rho^\beta(\mathbf{r}) = \sum_{\mathbf{k}}|\rho^\alpha(\mathbf{k})|^2+\sum_{\mathbf{k}}|\rho^\beta(\mathbf{k})|^2$$

is the electron density and the exchange correlation potential is:

$$V^{\text{xc}}[\rho(\mathbf{r})] = \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(\mathbf{r})}$$

The quantity $E_{\text{xc}}[\rho]$ is the exchange-correlation functional. Associated with the Kohn-Sham procedure is the Kohn-Sham determinant:

$$|I\rangle = |i\ldots j\ldots n|$$

In general, occupied orbitals of the electronic ground state are labeled with $i, j, k, \ldots$ and unoccupied ones with $a, b, c, \ldots$. In the one-electron approach, the initial, intermediate, and final states are constructed as:

$$|I\rangle = |i \ldots j \ldots n|$$
$$|V\rangle = |a \ldots j \ldots n|$$
$$|F\rangle = |a \ldots i \ldots n|$$

In the specific application at hand, orbital i is a core-orbital, e.g., the metal 1s derived orbital of a given absorber atom, a is an arbitrary unoccupied orbital (but most importantly an empty metal d-based orbital) and j is another core-level orbital of the same atom.

Rather than trying to obtain accurate total energies for all of these states, the simple approximations are adopted:

$$\omega_{jk} \approx \varepsilon_{\mathbf{k}} - \varepsilon_{\mathbf{j}}$$

This is of course fairly simplistic. In terms of time-dependent DFT, it can be shown that the orbital energy difference is a well-defined approximation to the state energy difference if no Hartree-Fock exchange is present in the DFT potential. For a semi-quantitative discussion it should be good enough since the substantial errors of the presently used DFT potentials prevent the calculations from predicting accurate transition energies anyways.

5.41.4 Non-Relativistic Treatment in 1-Electron Orbital Basis

In the non-relativistic treatment, states are expressed using 1-electron orbitals. The initial state is a Slater determinant of occupied orbitals i_σ, and the final state involves excitations to unoccupied orbitals a_σ, with $\sigma \in \alpha, \beta$ denoting the spin.

- **Non-Relativistic XAS**

The non-relativistic XAS cross section is:

$$\sigma_{\text{XAS}}(\omega_{ex}) = \frac{4\pi}{3c}\omega_{ex} \sum_{i_\sigma, a_\sigma} \sum_{\rho \in \{x,y,z\}} \frac{|\langle a_\sigma | m_\rho | i_\sigma \rangle|^2}{(\epsilon_{a_\sigma} - \epsilon_{i_\sigma} - \omega_{ex})^2 + \frac{1}{4}\Gamma^2}$$

or equivalently:

$$\sigma_{\text{XAS}}(\omega_{ex}) = \frac{4\pi}{3c}\omega_{ex} \sum_{i_\sigma, a_\sigma} \sum_{\rho \in \{x,y,z\}} |\langle a_\sigma | m_\rho | i_\sigma \rangle|^2 f(\omega_{a_\sigma i_\sigma}, \omega_{ex}, \Gamma)$$

- **Non-Relativistic XES**

The non-relativistic XES cross section is:

$$\sigma_{\text{XES}}(\omega_{em}) = \frac{4\pi}{3c}\omega_{em} \sum_{i_\sigma, j_\sigma} \sum_{\rho \in \{x,y,z\}} |\langle j_\sigma | m_\rho | i_\sigma \rangle|^2 f(\omega_{j_\sigma i_\sigma}, \omega_{em}, \Gamma)$$

- **Non-Relativistic Polarizability Tensor**

The non-relativistic polarizability tensor is:

$$\alpha_{\rho,\rho'}(\omega_{\text{ex}}, \omega_{\text{sc}}) = \sum_{i_\sigma} \sum_{j_\sigma} \sum_{a_\sigma} \langle a_\sigma | m_\rho | i_\sigma \rangle \langle i_\sigma | m_{\rho'} | j_\sigma \rangle \left[\frac{1}{\epsilon_{a_\sigma} - \epsilon_{i_\sigma} - \omega_{\text{ex}} - \frac{i}{2}\Gamma} + \frac{1}{\epsilon_{a_\sigma} - \epsilon_{j_\sigma} + \omega_{\text{sc}} - \frac{i}{2}\Gamma} \right]$$

Alternatively, if ω_{ex} is tuned to resonance with an intermediate state, the dominant term is:

$$|\alpha_{\rho\lambda}(\varepsilon_a^\sigma - \varepsilon_i^\sigma, \omega_{\text{sc}})|^2 = \sum_{i_\sigma} \sum_{j_\sigma} |\langle a_\sigma | m_\rho | i_\sigma \rangle \langle i_\sigma | m_\lambda | j_\sigma \rangle|^2 \left[\frac{\Gamma}{\left(\varepsilon_a^\sigma - \varepsilon_j^\sigma + \omega_{\text{sc}}\right)^2 + \frac{1}{4}\Gamma^2} \right]$$

Along the ω_{sc} axis, this is a resonance centered at $\varepsilon_j^\sigma - \varepsilon_a^\sigma$. This suggests providing a list of (i_σ, j_σ) with transition energies $E_{\omega_{\text{ex}}} = \varepsilon_a^\sigma - \varepsilon_i^\sigma$, $E_{\omega_{\text{sc}}} = \varepsilon_a^\sigma - \varepsilon_j^\sigma$, and values $|\langle a_\sigma | m_\rho | i_\sigma \rangle|^2 |\langle i_\sigma | m_\lambda | j_\sigma \rangle|^2$. Tools like orca_mapspc can read the linewidth and convolute peaks with a lineshape function, such as:

$$f(E_{VI}, E_{VF}, \omega_{ex}, \Delta\omega, \Gamma_V, \Gamma_F) = \left(\frac{\Gamma_V}{(E_{VI} - \omega_{ex})^2 + \frac{1}{4}\Gamma^2} \right) \left(\frac{\Gamma_F}{(E_{VF} - \Delta\omega)^2 + \frac{1}{4}\Gamma_F^2} \right)$$

This emphasizes that RIXS effectively measures L or M-edges through excitation at, e.g., the K-edge. Other lineshapes (Lorentzian, Gaussian, Voigt) are possible, as explained in sections (placeholder mapspc).

- **Non-Relativistic RIXS**

Hence the non-relativistic RIXS cross section becomes:

$$\begin{aligned}\sigma_{\text{RIXS}}(\omega_{ex},\omega_{sc}) =& \frac{8\pi}{9c^4}\omega_{ex}^3\omega_{sc} \\ &\times \sum_{i_\sigma}\sum_{j_\sigma}\sum_{a_\sigma} f(\Delta\omega_{a_\sigma i_\sigma},\Delta\omega_{j_\sigma i_\sigma},\omega_{ex},\Delta\omega,\Gamma_V,\Gamma_F) \\ &\times \sum_{\rho,\rho'\in\{x,y,z\}} |\langle a_\sigma|m_{\rho'}|i_\sigma\rangle\langle i_\sigma|m_\rho|j_\sigma\rangle|^2\end{aligned}$$

5.41.5 Inclusion of Relativistic Effects

There are two leading types of relativistic effects: scalar relativistic (spin free) relativistic effects and spin-orbit coupling.

- **Scalar Relativistic Effects**

Scalar relativistic effects can be included employing various schemes as described in *Relativistic Calculations* by modifying the one-electron part of the HamiltonianThe scalar relativistic effects show up as an attractive potential close to the nucleus. Hence s-, and p-orbitals are stabilized, d- and f-orbitals are destabilized. Concomitant with these effects, the radial functions of s- and p-orbitals will contract those of d-orbitals will expand. Hence, transition dipole moments will also be modified

- **Spin-Orbit Coupling (SOC)**

The spin-orbit coupling is much more problematic since the SOC is a two-electron operator that couples different spin-manifold and pure-spin multiplets. A good approximation is the spin-orbit meanfield (SOMF) treatment *The Spin-Orbit Coupling Operator* In this case, the SOC shows up as an effective one-electron operator of the form::

$H_{\text{SOMF}} = \sum_{\mathbf{k}} h^{\text{SOC}} s_{\mathbf{k}}$

- **Inclusion of SOC in 1-electron framework**

In \textit{ab initio} quantum chemistry one can use the Wigner–Eckart algebra in order to explicitly couple all M_s substates for states of total spin S. This corresponds roughly to LS coupling in the atomic case. For the SGS-DFT approach that we use here this is not an option and hence we are tied to jj coupling type treatments. The more rigorous version of this approach are two-component DFT approaches. Here we try a simple version of this more elaborate procedure. It consists of diagonalizing the Kohn–Sham and SOC operators in the basis of the converged non- or scalar relativistic spin-orbitals. Hence, the eigenvalue equation is:

$$\begin{bmatrix} \epsilon^\alpha & 0 \\ 0 & \epsilon^\beta \end{bmatrix} + \frac{1}{2}\begin{bmatrix} h_z^{\text{SOC}} & h_x^{\text{SOC}} - ih_y^{\text{SOC}} \\ h_x^{\text{SOC}} + ih_y^{\text{SOC}} & -h_z^{\text{SOC}} \end{bmatrix}\begin{bmatrix} \mathbf{p}_\alpha \\ \mathbf{p}_\beta \end{bmatrix} = \epsilon_n \begin{bmatrix} \mathbf{p}_\alpha \\ \mathbf{p}_\beta \end{bmatrix}$$

The matrix elements are in more detail:

$$\langle p_\alpha|F^\alpha + s_z h_z^{\text{SOC}}|q_\alpha\rangle = \delta_{pq}\epsilon_p^\alpha + i\frac{1}{2}L_{p_\alpha q_\alpha}$$

$$\langle p_\beta|F^\beta + s_z h_z^{\text{SOC}}|q_\beta\rangle = \delta_{pq}\epsilon_p^\beta - i\frac{1}{2}L_{p_\beta q_\beta}$$

$$\langle p_\alpha|F^\beta + s_x h_x^{\text{SOC}}|q_\beta\rangle = i\frac{1}{2}L_{p_\alpha q_\beta}$$

$$\langle p_\beta|F^\alpha + s_x h_x^{\text{SOC}}|q_\alpha\rangle = i\frac{1}{2}L_{p_\beta q_\alpha}$$

$$\langle p_\alpha|F^\beta + s_y h_y^{\text{SOC}}|q_\beta\rangle = -i\frac{1}{2}L_{p_\alpha q_\beta}$$

$$\langle p_\beta|F^\alpha + s_y h_x^{\text{SOC}}|q_\alpha\rangle = i\frac{1}{2}L_{p_\beta q_\alpha}$$

The matrix elements L_{pq}, are the spatial parts of the SOC operator and have the property that they are antisymmetric $L_{pq} = -L_{pq}$.

The diagonalization of this complex Hermitian matrix yields SOC corrected orbital energies and 'two component' orbitals that are linear combinations of the spin-free spin-orbitals:

$$|p\rangle = \sum_{r_\alpha} V_{r_\alpha p}|r_\alpha\rangle + \sum_{r_\beta} V_{r_\beta p}|r_\beta\rangle$$

From these orbitals one readily calculates the transition moments:

$$\langle p|\mathbf{m}|p\rangle = \sum_{r_\alpha r_\alpha} V^*_{r_\alpha p} V_{s_\alpha q}\langle r_\alpha|\mathbf{m}|s_\alpha\rangle + \sum_{r_\beta r_\beta} V^*_{r_\beta p} V_{s_\beta q}\langle r_\beta|\mathbf{m}|s_\beta\rangle$$

where the V's are the eigenfunctions of the $\mathbf{F} + \mathbf{h}^{\text{SOC}}$ matrix. With these transition moments and orbital energies all XAS, XES and RIXS is computed as before; one simply has the spin-orbit label p.

5.41.6 Computations of X-ray Spectroscopic Properties

In the following examples we present a variety of X-ray spectroscopic employing the conventional BP86 in conjuctuon with the X2C relativstic hamiltonian and the reseoctive triple-zeta basis set. However in principle any relativistic scheme can be used instead *Basis Sets in Relativistic Calculations*

We choose to describeexplicitely the X-ray computed spectra of the octagedral Fe^{III} complex $[Fe^{III}Cl_6]^{3-}$.

```
!BP86 X2C x2c-TZVPall-s x2c/J

%xes
 #-------------------------------
 #    General Definitions
 #-------------------------------
 CoreOrb 0,0
 OrbOp 0,1
 CoreOrbSOC 0,1
 #-------------------------------
 #    Spectroscopic Properties
 #-------------------------------
 DoXAS true
 DoSOC true
 #-------------------------------
 #    Spectra Intensities
 #-------------------------------
 DoDipoleLength true
 DoFullSemiclassical true
end

* xyz -1 6
Fe      -0.008368229      0.000000000      0.012210784
Cl      -1.375773751     -1.142123897     -0.186609497
Cl      -0.530592683      1.387798468      1.019718298
Cl       0.513856225      0.592852206     -1.597007114
Cl       1.359037292     -0.838526778      0.812741448
*
```

As the core of the SCF-XES or SGS-XES method involves the derivation of all important quantities in orbital and spin-orbital basis for computing the K-edge Fe XAS as well as the resepctive Mainline XES and the corresponding (Valence to Core) VtC-XES spectra one chooses the list of the desired core orbitals here the Fe_{1s} with the resepctive Orbital (0 and 1) and SOC Orbital operators (here 0 and 1).

```
 CoreOrb 0,0
 OrbOp 0,1
 CoreOrbSOC 0,1
```

And we compute both the:

- K_α: Fe $2p \rightarrow$ Fe $1s$ (Mainline XES)
- $K_{\beta_{1,3}}$: Fe $3p \rightarrow$ Fe $1s$ (Mainline XES)
- $K_{\beta_{2,5}}$ / $K_{\beta'}$: (Cl $2p$ + Fe $3d$) $\rightarrow$ Fe $1s$ (VtC-XES)

besides the K-edge Fe XAS spectra

In the non relativistic section one then obtains the XAS and XES spectra within the ORCA's One-Photon Spectroscopy framework

```
-------------------------------------
CALCULATION OF X-RAY EMISSION SPECTRUM
-------------------------------------
Reading and Transforming integrals                ... Generating transition␣
↪densities between Orbitals         ... done
Reading integrals                                                   ...
Reading the Dipole integrals                                        ... done

Calculating the Full-semiclassical Transition moment on transition block:0        .
↪..
10% done
20% done
30% done
40% done
50% done
60% done
70% done
80% done
90% done
100% done
All done in  (       29.14 sec)
-------------------------------------------------------------------
Using One-Photon Spectroscopy Tool
-------------------------------------------------------------------


-----------------------------------------------------------------------------------
↪-----------------
              X-RAY   EMISSION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)    (nm)                 (au**2)    (au)    ␣
↪ (au)       (au)
-----------------------------------------------------------------------------------
↪-----------------
  1a-a  ->  0a-a   4243.560625 34226628.2     0.3   0.000000000   0.00000   0.
↪00000  -0.00000  -0.00000
  2a-a  ->  0a-a   4243.562891 34226646.5     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  3a-a  ->  0a-a   4243.563723 34226653.2     0.3   0.000000000   0.00000  -0.
↪00000   0.00000  -0.00000
  4a-a  ->  0a-a   4243.563845 34226654.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000  -0.00000
  1b-a  ->  0b-a   4243.609379 34227021.5     0.3   0.000000000   0.00000   0.
↪00000  -0.00000  -0.00000
  2b-a  ->  0b-a   4243.611639 34227039.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  3b-a  ->  0b-a   4243.612467 34227046.4     0.3   0.000000000   0.00000   0.
↪00000  -0.00000   0.00000
  4b-a  ->  0b-a   4243.612590 34227047.4     0.3   0.000000000   0.00000   0.
```

(continues on next page)

(continued from previous page)

```
↪00000   0.00000  -0.00000
  5a-a  ->  0a-a   6174.919612 49804090.7     0.2   0.000000000   0.00000  -0.
↪00000   0.00000   0.00000
  5b-a  ->  0b-a   6176.509051 49816910.3     0.2   0.000000000   0.00000  -0.
↪00000   0.00000   0.00000
  6a-a  ->  0a-a   6296.393596 50783844.5     0.2   0.101316564   0.00066  -0.
↪02354   0.00780  -0.00648
  7a-a  ->  0a-a   6296.393628 50783844.8     0.2   0.101316573   0.00066  -0.
↪00668   0.00037   0.02474
  8a-a  ->  0a-a   6296.393675 50783845.1     0.2   0.101316586   0.00066  -0.
↪00762  -0.02441  -0.00169
  6b-a  ->  0b-a   6297.611741 50793669.5     0.2   0.101563788   0.00066   0.
↪02425  -0.00783   0.00303
  7b-a  ->  0b-a   6297.611770 50793669.7     0.2   0.101563797   0.00066  -0.
↪00317   0.00003   0.02546
  8b-a  ->  0b-a   6297.611810 50793670.1     0.2   0.101563807   0.00066  -0.
↪00777  -0.02443  -0.00094

...

-----------------------------------------------------------------------------
              X-RAY   EMISSION SPECTRUM VIA FULL SEMI-CLASSICAL FORMULATION
-----------------------------------------------------------------------------
     Transition      Energy     Energy  Wavelength   fosc(FFMIO)
                      (eV)      (cm-1)     (nm)
-----------------------------------------------------------------------------
  1a-a  ->  0a-a   4243.560625 34226628.2     0.3     0.00000000000264
  2a-a  ->  0a-a   4243.562891 34226646.5     0.3     0.00000000000270
  3a-a  ->  0a-a   4243.563723 34226653.2     0.3     0.00000000000328
  4a-a  ->  0a-a   4243.563845 34226654.2     0.3     0.00000000000163
  1b-a  ->  0b-a   4243.609379 34227021.5     0.3     0.00000000000264
  2b-a  ->  0b-a   4243.611639 34227039.7     0.3     0.00000000000270
  3b-a  ->  0b-a   4243.612467 34227046.4     0.3     0.00000000000327
  4b-a  ->  0b-a   4243.612590 34227047.4     0.3     0.00000000000163
  5a-a  ->  0a-a   6174.919612 49804090.7     0.2     0.00000000000007
  5b-a  ->  0b-a   6176.509051 49816910.3     0.2     0.00000000000006
  6a-a  ->  0a-a   6296.393596 50783844.5     0.2     0.10239197992333
  7a-a  ->  0a-a   6296.393628 50783844.8     0.2     0.10239199025533
  8a-a  ->  0a-a   6296.393675 50783845.1     0.2     0.10239200379726
  6b-a  ->  0b-a   6297.611741 50793669.5     0.2     0.10265622291301
  7b-a  ->  0b-a   6297.611770 50793669.7     0.2     0.10265623313526
  8b-a  ->  0b-a   6297.611810 50793670.1     0.2     0.10265624471656

----------------------------------------
CALCULATION OF X-RAY ABSORPTION SPECTRUM
----------------------------------------
Reading integrals                                                  ...
Reading the Dipole integrals                                       ... done

Calculating the Full-semiclassical Transition moment on transition block:0        .
↪..
10% done
20% done
30% done
40% done
50% done
60% done
70% done
80% done
90% done
100% done
```

(continues on next page)

(continued from previous page)

```
All done in  (      105.64 sec)
-----------------------------------------------------------------
Using One-Photon Spectroscopy Tool
-----------------------------------------------------------------

-------------------------------------------------------------------------------
↪-----------------
              X-RAY ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------
↪-----------------
     Transition      Energy     Energy  Wavelength fosc(D2)      D2        DX     ␣
↪  DY        DZ
                      (eV)      (cm-1)    (nm)                 (au**2)    (au)    ␣
↪ (au)      (au)
-------------------------------------------------------------------------------
↪-----------------
  0b-a  -> 45b-a   6988.978848 56369921.9     0.2   0.000000000   0.00000   0.
↪00000   0.00000  -0.00000
  0b-a  -> 46b-a   6988.978999 56369923.1     0.2   0.000000000   0.00000   0.
↪00000  -0.00000  -0.00000
  0b-a  -> 47b-a   6990.167972 56379512.8     0.2   0.000111632   0.00000  -0.
↪00045   0.00053   0.00040
  0b-a  -> 48b-a   6990.168059 56379513.5     0.2   0.000111638   0.00000  -0.
↪00048  -0.00060   0.00026
  0b-a  -> 49b-a   6990.168100 56379513.8     0.2   0.000111637   0.00000  -0.
↪00047   0.00010  -0.00065
  0a-a  -> 50a-a   6992.570956 56398894.2     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0b-a  -> 50b-a   6992.689756 56399852.4     0.2   0.000000000   0.00000  -0.
↪00000   0.00000   0.00000
  0a-a  -> 51a-a   6995.905926 56425792.5     0.2   0.000037002   0.00000  -0.
↪00036  -0.00006   0.00029
  0a-a  -> 52a-a   6995.906125 56425794.1     0.2   0.000036975   0.00000   0.
↪00006  -0.00046  -0.00003
  0a-a  -> 53a-a   6995.906494 56425797.1     0.2   0.000037064   0.00000  -0.
↪00029  -0.00002  -0.00036
  0a-a  -> 54a-a   6996.058845 56427025.9     0.2   0.000006536   0.00000   0.
↪00010  -0.00009   0.00014
  0a-a  -> 55a-a   6996.059201 56427028.8     0.2   0.000006569   0.00000   0.
↪00016   0.00010  -0.00005
  0a-a  -> 56a-a   6996.059311 56427029.7     0.2   0.000006619   0.00000  -0.
↪00004   0.00014   0.00013
  0b-a  -> 51b-a   6996.206394 56428216.0     0.2   0.000014342   0.00000   0.
↪00021  -0.00015  -0.00013

-----------------------------------------------------------------------------
              X-RAY ABSORPTION SPECTRUM VIA FULL SEMI-CLASSICAL FORMULATION
-----------------------------------------------------------------------------
     Transition      Energy     Energy  Wavelength   fosc(FFMIO)
                      (eV)      (cm-1)    (nm)
-----------------------------------------------------------------------------
  0b-a  -> 45b-a   6988.978848 56369921.9     0.2     0.00000231997701
  0b-a  -> 46b-a   6988.978999 56369923.1     0.2     0.00000232027335
  0b-a  -> 47b-a   6990.167972 56379512.8     0.2     0.00011661916478
  0b-a  -> 48b-a   6990.168059 56379513.5     0.2     0.00011662652902
  0b-a  -> 49b-a   6990.168100 56379513.8     0.2     0.00011662384321
  0a-a  -> 50a-a   6992.570956 56398894.2     0.2     0.00000000000014
  0b-a  -> 50b-a   6992.689756 56399852.4     0.2     0.00000000000008
  0a-a  -> 51a-a   6995.905926 56425792.5     0.2     0.00004544364720
  0a-a  -> 52a-a   6995.906125 56425794.1     0.2     0.00004540088165
  0a-a  -> 53a-a   6995.906494 56425797.1     0.2     0.00004552596978
```

(continues on next page)

(continued from previous page)

```
  0a-a  -> 54a-a   6996.058845 56427025.9     0.2     0.00001087977536
  0a-a  -> 55a-a   6996.059201 56427028.8     0.2     0.00001092658019
  0a-a  -> 56a-a   6996.059311 56427029.7     0.2     0.00001099716202
  0b-a  -> 51b-a   6996.206394 56428216.0     0.2     0.00001669184884
```

The program the proceeds to generate the SOC corrected orbitals

```
----------------------------------------
XES/XAS SPECTRUM WITH SPIN ORBIT COUPLING
----------------------------------------

Reading the SOC integrals             ... done
Forming the SOC Hamiltonian           ... done
Diagonalizing the SOC Hamiltonian     ... done
-------------------------------
SOC CORRECTED OCCUPIED ORBITALS
-------------------------------
 E[   0] = -6987.772 eV: ( -0.512 +  -0.859 *i)|   0a>
 E[   1] = -6987.756 eV: (  0.002 +  -1.000 *i)|   0b>
 E[   2] = -2744.212 eV: (  0.383 +  -0.924 *i)|   1a>
 E[   3] = -2744.209 eV: ( -0.772 +  -0.636 *i)|   2a>
 E[   4] = -2744.208 eV: ( -0.952 +   0.307 *i)|   3a>
 E[   5] = -2744.208 eV: ( -0.923 +   0.384 *i)|   4a>
 E[   6] = -2744.147 eV: ( -0.001 +  -1.000 *i)|   1b>
 E[   7] = -2744.144 eV: (  0.001 +  -1.000 *i)|   2b>
 E[   8] = -2744.144 eV: (  0.244 +   0.970 *i)|   3b>
 E[   9] = -2744.143 eV: (  0.025 +  -1.000 *i)|   4b>
 E[  10] =  -812.853 eV: (  0.512 +  -0.859 *i)|   5a>
 E[  11] =  -811.247 eV: (  0.163 +   0.987 *i)|   5b>
 E[  12] =  -698.833 eV: (  0.548 +  -0.181 *i)|   6a> (  0.155 +  -0.009 *i)|  ␣
↪7a> (  0.177 +   0.568 *i)|   8a> ( -0.534 +  -0.000 *i)|   7b>
 E[  13] =  -698.431 eV: (  0.072 +  -0.139 *i)|   6a> ( -0.275 +   0.529 *i)|  ␣
↪7a> (  0.393 +  -0.388 *i)|   6b> (  0.392 +   0.394 *i)|   8b>
 E[  14] =  -687.512 eV: ( -0.671 +   0.134 *i)|   6a> ( -0.165 +   0.082 *i)|  ␣
↪7a> (  0.150 +   0.689 *i)|   8a>
 E[  15] =  -687.067 eV: ( -0.092 +   0.176 *i)|   6a> (  0.350 +  -0.673 *i)|  ␣
↪7a> (  0.309 +  -0.305 *i)|   6b> (  0.308 +   0.310 *i)|   8b>
 E[  16] =  -686.654 eV: ( -0.349 +   0.116 *i)|   6a> ( -0.113 +  -0.362 *i)|  ␣
↪8a> ( -0.837 +  -0.000 *i)|   7b>
 E[  17] =  -686.266 eV: (  0.500 +   0.493 *i)|   6b> (  0.499 +  -0.501 *i)|  ␣
↪8b>
```

in following all spectra are recomputed in the OPS framework in the presence of SOC

```
Generating transition densities between SOC_Corrected_Orbitals         ... done
-----------------------------------------------------
CALCULATION OF SOC CORRECTED X-RAY EMISSION SPECTRUM
-----------------------------------------------------
Reading integrals                                                  ...
Reading the Dipole integrals                                       ... done

Calculating the Full-semiclassical Transition moment on transition block:0        .
↪..
10% done
20% done
30% done
40% done
50% done
60% done
70% done
80% done
```

(continues on next page)

(continued from previous page)

```
90% done
100% done
All done in  (       58.75 sec)
-------------------------------------------------------------------
Using One-Photon Spectroscopy Tool
-------------------------------------------------------------------


-------------------------------------------------------------------------------------
↪---------------------
   SPIN-ORBIT X-RAY   EMISSION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-------------------------------------------------------------------------------------
↪---------------------
      Transition          Energy     Energy  Wavelength fosc(D2)      D2       |DX|␣
↪     |DY|      |DZ|
                           (eV)      (cm-1)    (nm)                 (au**2)    (au)␣
↪     (au)      (au)
-------------------------------------------------------------------------------------
↪---------------------
     2-a  ->  0-a      4243.560625 34226628.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     2-a  ->  1-a      4243.544569 34226498.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     3-a  ->  0-a      4243.562891 34226646.5     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     3-a  ->  1-a      4243.546835 34226517.0     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     4-a  ->  0-a      4243.563723 34226653.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     4-a  ->  1-a      4243.547667 34226523.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     5-a  ->  0-a      4243.563845 34226654.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     5-a  ->  1-a      4243.547790 34226524.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     6-a  ->  0-a      4243.625435 34227151.0     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     6-a  ->  1-a      4243.609379 34227021.5     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     7-a  ->  0-a      4243.627694 34227169.2     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     7-a  ->  1-a      4243.611639 34227039.7     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     8-a  ->  0-a      4243.628523 34227175.9     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     8-a  ->  1-a      4243.612467 34227046.4     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     9-a  ->  0-a      4243.628646 34227176.9     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
     9-a  ->  1-a      4243.612590 34227047.4     0.3   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
    10-a  ->  0-a      6174.919612 49804090.7     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
    10-a  ->  1-a      6174.903556 49803961.2     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
    11-a  ->  0-a      6176.525107 49817039.8     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
    11-a  ->  1-a      6176.509051 49816910.3     0.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
    12-a  ->  0-a      6288.939050 50723719.5     0.2   0.072768392   0.00047   0.
↪01537   0.01537   0.00000
    12-a  ->  1-a      6288.922994 50723590.0     0.2   0.029738152   0.00019   0.
```

(continues on next page)

(continued from previous page)

```
↪00000   0.00000   0.01389
    13-a  ->  0-a      6289.340683 50726958.9     0.2   0.039112694   0.00025   0.
↪00000   0.00000   0.01593
    13-a  ->  1-a      6289.324627 50726829.4     0.2   0.063473047   0.00041   0.
↪01435   0.01435   0.00000
    14-a  ->  0-a      6300.260175 50815030.6     0.2   0.100751569   0.00065   0.
↪01807   0.01807   0.00000

...

and so on for the rest of the spectra tables
```

All Spectra are generated using orca_mapspc with [Options] for XAS, XES, XASFFMIO and XESSFFMIO, including electric dipole (ED), with full field matter interaction operator (FFMIO) with and without SOC:

```
   orca_mapspc FileName.out [Options] -x06000 -x18000 -w2.0 -n10000
```

The computed XAS spectra in the electric dipole approximation EDA, and employing the FFMIO semicalssical operator with and wthout SOC are shown in Figure Fig. 5.73.

Fig. 5.73: Comparison of the calculated SGS-DFT Fe K-edge XAS spectra of $[FeCl_4]^{1-}$. within the EDA (blue) and FFMIO (red) approximations.

In following the computed K_α: Mainline XES spectra employing the FFMIO semicalssical operator with and wthout SOC are shown in Figure Fig. 5.74. In the case of SOC one observes explicitely the SOC splitting owning to the 2p core-hole in the 2p $\rightarrow$ Fe $1s$ (Mainline XES) electron decay process.

Fig. 5.74: Comparison of the calculated SGS-DFT K_α: Mainline XES spectra of $[FeCl_4]^{1-}$. employing the FFMIO semicalssical operator with SOC (red) and without SOC (blue) corrections.

in this state to orbital approach where in principle every orbital excitation constitute a state withing the SGS-DFT framework, in the same calculation setup, one also obtains in addition all accesible electron decay processes. This means that in addition to the
$K_{\beta_{1,3}}$: Fe $3p \rightarrow$ Fe $1s$ (Mainline XES) as well as the $K_{\beta_{2,5}}$ / $K_{\beta'}$: (Cl $2p$ + Fe $3d$) $\rightarrow$ Fe $1s$ (VtC-XES) maybe accessed as is shown in Figure Fig. 5.75.

Fig. 5.75: Calculated SGS-DFT K_β: Mainline XES (red) and VtC XES (blue) spectra of $[FeCl_4]^{1-}$. employing the FFMIO semicalssical operator with SOC corrections

In the same framework one can also compute Simple RIXS spectra in the Direct RIXS fashion… To become available in the next minor release. For more information about the RIXS spectra theory and generation please have a look in section: *Resonant Inelastic Scattering Spectroscopy*

The entire keyword setup in the %xes block of SCF-XES or SSS-DFT method is given below

```
%xes
 #-------------------------------
 #    Core Orbital Definitions
 #-------------------------------
 CoreOrb 0,0 # list of core orbitals (alpha,-beta)
 OrbOp 0,1 # operators for core-orbitals (0=alpha,1=beta)
 CoreOrbSOC 0,1 # core orbitals for SOC calculations
 #-------------------------------
 #    RIXS Orbital Definitions
 #-------------------------------
 RIXSOrb    6,7,8,6,7,8       # RIXS final orbitals
 RIXSOrbOp  0,0,0,1,1,1       # operators for these orbitals
 RIXSOrbSOC 12,13,14,15,16,17 # same for SOC
 #-------------------------------
 #    Virtual Orbital Definitions
 #-------------------------------
 MaxNVirt 200 #max number of virtual Orbitals
 #-------------------------------
 #    Spectroscopic Properties
 #-------------------------------
 DoXAS true # Do calculate XAS spectra in addition to XES
 DoSOC true # Do calculate SOC corrected  spectra
 DoRIXS true # Do calculate RIXS  spectra
 #-------------------------------
 #    Spectra Intensities
```

(continues on next page)

(continued from previous page)

```
 #-------------------------------
 DoDipoleLength true # Calculate Spectra in Electric Dipole EDA (Length/Velocity)
 DoFullSemiclassical true # Calculate Spectra in Semiclassical FFMIO framework
end
```

Note

It is in general not recomended to compute X-ray spectra with complex multiplet structure via the SGS-DFT approach. This involves e.g. metal L/M-edge XAS spectra, K_α: Fe $2p \rightarrow$ Fe $1s$ and $K_{\beta_{1,3}}$: Fe $3p \rightarrow$ Fe $1s$ (Mainline XES) as well as RIXS spectra. The computation of such spectroscopic problem is preferably tackled on the basis of wavefunction methodologies:

- ROCIS and CASCI/RASCI (in combination with NEVPT2) or multirefernce approaches (e.g. MREOM-CC) and MCRPA are able to tackle a variety of such complex spectroscopes, see discussions in *Transition Metal L-Edges with ROCIS or DFT/ROCIS*, *Resonant Inelastic Scattering Spectroscopy*, *Core Excited Spectra: CAS-CI/RAS-CI XES*, *Excited States via MCRPA*
- In a similar framework TD-DFT has been proven instrumental in computing metal and ligand K-edge spectra, see discussions in *Excited States via RPA, CIS, TD-DFT and SF-TDA*.
- Core level spectroscopy spectra (presently Metal/Ligand K-edge XAS) can also be computed with couple-cluster (EOM-CC, STEOM-CC) methods see discussions in *Core-Level Spectroscopy with Coupled Cluster Methods*

CHAPTER

SIX

MULTISCALE SIMULATIONS

6.1 Overview on ORCA's Multiscale Implementation

ORCA features five primary methods suitable for diverse applications such as proteins, DNA, large molecules, explicit solvation, and crystalline solids:

- *Additive QM/MM* for proteins and DNA/RNA.
- *QM1/QM2* (2-layered ONIOM) for small proteins, RNA, large molecules and explicit solvation.
- *QM1/QM2/MM* (3-layered ONIOM) for proteins and DNA/RNA.
- *MOL-CRYSTAL-QMMM* for molecular crystals.
- *IONIC-CRYSTAL-QMMM* for semiconductors and insulators.

All methods integrate seamlessly with other ORCA modules, simplifying the preparation and execution of multiscale calculations.

The multiscale features are optimally connected to all other modules and tools available in ORCA allowing the user to handle multiscale calculations from a QM-centric perspective in a simple and efficient way, with a focus on simplifying the process to prepare, set up and run multiscale calculations. Since version 6.1 ORCA provides an automated tool for setting up QM and active regions, which is described *here*.

From the input side all methods share a common set of concepts and keywords, which will be outlined in this section. In subsequent sections the different methods are described and further input options are discussed.

6.1.1 Combining Multiscale Features with other ORCA Features

The multiscale features can be used together with all other possible ORCA methods:

- **Single Point Calculations**: Use all kinds of available electronic structure methods as QM method.
- **Optimization**: Use all kinds of *geometry optimizations* with all kinds of constraints, TS optimization, relaxed surface scans, and the ScanTS feature. Use the L-Opt and L-OptH features including the combination of all kinds of fragment optimizations (fix fragments, relax fragments, relax only specific elements in fragments, treat a fragment as a rigid body).
- **Transition States and Minimum Energy Paths**: Use all kinds of *Nudged-Elastic Band calculations* Nudged-Elastic Band calculations (Fast-NEB-TS, NEB, NEB-CI, NEB-TS, including their ZOOM variants) and Intrinsic Reaction Coordinate calculations. (not implemented for `MOL-CRYSTAL-QMMM` and `IONIC-CRYSTAL-QMMM`)
-
- **Conformational Searches**: Conformational Searches can be run with ORCA's *GOAT* module.
- **Frequency Calculations**: Use regular *frequency calculations*. If required, ORCA automatically switches on the *Partial Hessian Vibrational Analysis* (PHVA) [531] calculation. (not tested for `IONIC-CRYSTAL-QMMM`)

- **Molecular Dynamics**: Use the *Molecular Dynamics* (MD) module for Born-Oppenheimer MD (BOMD) with QM/MM in combination with all kinds of electronic structure methods. (not implemented for `MOL-CRYSTAL-QMMM` and `IONIC-CRYSTAL-QMMM`)
- **Property Calculation**: All kinds of regular property calculations are available. For electrostatic embedding the electron density is automatically perturbed by the surrounding point charges.
- **Excited State Calculations**: Use all kinds of *excited state calculations* (TD-DFT, EOM, single point calculations, optimizations, frequencies).

6.1.2 Required Settings for a Multiscale Simulation

In the following, the basic concepts are introduced.

- **QM atoms**: The user can define the QM region either directly, or via flags in a pdb file. See *QM and Active Region Definition*. For an automated buildup
- of the QM region with miminal user input see *here*.
- **Active Atoms**: The user can choose an active region, e.g. for geometry optimizations the atoms that are optimized, for a frequency calculation the atoms that are allowed to vibrate for the PHVA, or for an MD run the atoms that are propagated. More background is provided *here*. Input options on how to define the active region in an *automated* way or in an *explicit* way are provided below.

A minimal user input for an ONIOM2 type calculation looks like this.

```
!QM/XTB
%qmmm
  QMCoreAtoms {7 12} end
  ActiveCoreAtoms {7 12} end
end
*xyz 0 1
...
```

Other aspects that often need to be addressed for multiscale simulations are treated automatically and with reasonable defaults- which can, of course, be controlled by the user. The **boundary treatment**, i.e. bonds that have to be cut between QM and MM region, and the treatment of overpolarization is covered in the paragraph *Boundary Region*. Available **embedding types** are discussed in section *Embedding Types*. **QM2 atoms**, which define the low-level region in 3-layer ONIOM calculations, are discussed in section *QM2 Atoms*.

Detailed information on the different available input and runtime options and additionally available keywords (see *Additional Keywords*) are given below.

> **Note**
>
> In order to check the multiscale setup done by ORCA, without the need to run the entire simulation, you can invoke the keyword `! QMMMSetup`. This makes sure that ORCA stops after the multiscale preparation. The information on the QMMM setup is printed in the output, and a file with the QM region including link atoms is stored (`basename_QM.xyz`).

6.1.3 QM and Active Region Definition

QM and active region atoms can be defined in three different ways:

- Explicit definition of all atom indices.
- Defining the most important atoms (the `core` atoms) and ORCA determines the full QM and active region.
- Definition in a pdb file.

Explicit Definition of QM and Active Region

The QM and active region can be directly defined in the input as shown in the following example:

```
%qmmm
 QMAtoms {0 1 2 27 28} end      # lists of atoms (counting starts from 0) or
 ActiveAtoms {0:5 16 21:30} end # ranges of atoms (range defined via start:end)
end
*pdbfile 0 1 ubq.pdb
```

Automated Definition of QM and Active Region

A standard approach in the QMMM scientific literature for defining QM and active regions involves:

Identifying Important Atoms or Functional Groups

Examples include:

- Atoms involved directly in bond formation or breakage.
- Functional groups essential for spectroscopy.
- Metal ions central to catalytic activities.
- Entire molecules or ligands binding to proteins.
- *Additional criteria as necessary.*

Including Surrounding Atoms

- Include all atoms within a defined spatial distance around these important atoms.

Applying Rules to Avoid Artifacts

Specifically ensure that:

- The boundary between QM and MM regions cuts through a non-polarized single bond.
- The topological distance between two QM-MM cuts (the MM gap) is sufficiently large,
- preventing overly close placement of link atoms.

Automated procedure in ORCA 6.1

In ORCA 6.1, an automatic scheme has been developed to manage this multistep process. Users specify only key atoms (**core** atoms) that define the core part of the QM and active regions.

ORCA requires:

- Knowledge of the system's **topology**.
- Information about **functional groups** within the system.

Using the newly introduced *fragmentation procedure*, ORCA automatically determines this information at the start of the calculation. Functional groups are stored as separate fragments based on the user-selected fragmentation procedure.

Steps of the Automated Procedure (Illustrated for QM Region; analogous steps apply for active regions)

1. Accept user-defined input `QMCoreAtoms`.
2. If `QMCore_Type` is `Fragments`, extend `QMCoreAtoms` to entire fragments, producing an updated `QMCoreAtoms` set.
3. Determine all atoms within `QMCore_Extension` Å around `QMCoreAtoms`, yielding `QMAtoms`.
4. If `QMCoreExtension_Type` is `Fragments`, extend `QMAtoms` to entire fragments, updating the set again.
5. Include any explicitly user-defined atoms (`QMAtoms`) in the automatically generated set.
6. If `CheckAutoFragForQMGaps` is `true`, verify no topologically close QM atoms are separated by fewer MM atoms than `QMGap_MinLength`. If so, add necessary connecting fragments to `QMAtoms`.
7. If `CheckAutoFragBoundaries` is `true`, ensure that QM-MM boundaries do not cut polarized or problematic bonds.
8. Repeat steps 6 and 7 until no further fragments need addition.

> **Note**
>
> You can add the keyword `QMMMSetup` to your input. This allows you to check the autogenerated QM region (`basename_QM.xyz`) and verify the charge and multiplicity required for the actual multiscale calculation. that is needed for the actual multiscale calculation.

The automated QM and active region setup can be defined in the input as shown in the following example:

```
! QM/XTB
! QMMMSetup # just do the preparation and stop
%qmmm
 QMCoreAtoms {2032} end
 ActiveCoreAtoms {2032} end
 frag
   FragProc NABackbone, NASideChains, FunctionalGroups, Extend, Connectivity
 end
end

*xyz 0 1
```

Fig. 6.1: QM region for QMCoreAtom 2032. Left is the QM region in ball-and-stick style as part of the full system in cartoon style, right is the QM region with link atoms.

In the above example, which is a ligand bound to RNA, we first define the fragments using the *fragmentator*) using RNA specific groups and functional groups (for the ligand), and then build the QM and active region around the ligand (atom 2032 is an atom of the ligand).

```
! QMMMSetup # just do the preparation and stop
! QMMM
%qmmm
  ORCAFFFilename "reactant.ORCAFF.prms"
  QMCoreAtoms {183 2745} end
  ActiveCoreAtoms {183 2745} end
  frag
    FragProc SEQBACKBONE, AASCFINEGRAINED, FUNCTIONALGROUPS, EXTEND, Connectivity
    UseTopology true
    TopolFile "reactant.ORCAFF.prms"
  end
end
*pdbfile 0 1 reactant.pdb
```

Fig. 6.2: QM region for QMCoreAtoms 183 and 2745. Left is the QM region in ball-and-stick style as part of the full system in cartoon style, right is the QM region with link atoms.

In the above example, involving a covalent ligand bound within a protein active site, we first define the fragments using the *fragmentator*, specifying protein-specific groups and functional groups for ligands and cofactors. We then construct the QM and active regions around the two critical atoms directly involved in the covalent binding process: atom `183` from the Cys reside, and atom `2745` from the covalently binding ligand.

Table 6.1: Options to automatically define the QM and active region.

Keyword	Options	Description
QMCOREATOMS	{ints ..}	Defines the core atoms of the QM region. Single indices can be given as int int. Ranges of atom indices can be provided as int:int. All these can be combined.
ACTIVECOREATO	{ints	Defines the core atoms of the active region.
Frag	As defined *here*	
QMCore_Type	Fragm	(Default) Take as QM core the atoms that the user defines plus all atoms of the fragments which they belong to.
	Atoms	Take as QM core only the atoms that the user defines.
QMCoreExtensi	Fragm	(Default) Take as QMAtoms the atoms that belong to fragments which have at least one atom which has a distance of less than QMCore_Extension to any of the QMCoreAtoms.
	Atoms	Take as QMAtoms the atoms that have a distance of less than QMCore_Extension to any of the QMCoreAtoms.
QMCore_Extens	real	Distance by how much the QM core is extended to yield QMAtoms. Default 3.
CheckAutoFrag	true/f	Check whether two QM atoms show a QM gap, i.e. which are topologically connected by more than QMGap_MinLength MM atoms and if so, add the connecting fragments to QMAtoms.
QMGap_MinLeng	0 to 6	Topological distance between two QM atoms which determines whether the connecting MM atoms are considered as a QM gap. Default 3
CheckAutoFrag	true/f	Check whether a QM-MM bond is an unpolar single bond. If not, add the MM atom's fragment to QMAtoms.
ActiveCore_Ty	Fragm	(Default) Take as active core the atoms that the user defines plus all atoms of the fragments which they belong to.
	Atoms	Take as active core only the atoms that the user defines.
ActiveCoreExt	Fragm	(Default) Take as ActiveAtoms the atoms that belong to fragments which have at least one atom which has a distance of less than ActiveCore_Extension to any of the ActiveCoreAtoms.
	Atoms	Take as ActiveAtoms the atoms that have a distance of less than ActiveCore_Extension to any of the ActiveCoreAtoms.
ActiveCore_Ex	real	Distance by how much the active core is extended to yield ActiveAtoms. Default 5.
CheckAutoFrag	true/f	Check whether two active atoms show a gap, i.e. they are topologically connected by more than QMGap_MinLength non-active atoms and if so, add the connecting fragments to ActiveAtoms.
CheckAutoFrag	true/f	Check whether a active-nonactive bond is an unpolar single bond. If not, add the non-active atom's fragment to ActiveAtoms.

Note

Automated setup can be combined with explicit definition of QM or active atoms via QMAtoms and ActiveAtoms.

Definition of QM and Active Region via pdb file

The QM and active region can be defined in the pdb file as shown in the following example:

```
%qmmm
 Use_QM_InfoFromPDB true      # get the QMAtoms and ActiveAtoms definition from
 Use_Active_InfoFromPDB true # the pdb file. The default for both is false.
                              # If InfoFromPDB is set to true, the QMAtoms or
                              # ActiveAtoms input is ignored
end
*pdbfile 0 1 ubq.pdb
```

If `Use_QM_InfoFromPDB` or `Use_Active_InfoFromPDB` is set to true, a pdb file should be used for the structural input. QM atoms are defined via 1 in the occupancy column, MM atoms via 0. Active atoms are defined via 1 in the B-factor column, non-active atoms via 0.

```
ubq.pdb:
...
ATOM    327  N   ASP A  21      29.599  18.599   9.828  0.00  0.00      P1   N
ATOM    328  HN  ASP A  21      29.168  19.310   9.279  0.00  0.00      P1   H
ATOM    329  CA  ASP A  21      30.796  19.083  10.566  0.00  1.00      P1   C
ATOM    330  HA  ASP A  21      31.577  18.340  10.448  0.00  1.00      P1   H
ATOM    331  CB  ASP A  21      31.155  20.515  10.048  2.00  1.00      P1   C
ATOM    332  HB1 ASP A  21      30.220  21.082   9.865  2.00  1.00      P1   H
ATOM    333  HB2 ASP A  21      31.754  21.064  10.801  2.00  1.00      P1   H
ATOM    334  CG  ASP A  21      31.923  20.436   8.755  1.00  1.00      P1   C
ATOM    335  OD1 ASP A  21      32.493  19.374   8.456  1.00  1.00      P1   O
ATOM    336  OD2 ASP A  21      31.838  21.402   7.968  1.00  1.00      P1   O
ATOM    337  C   ASP A  21      30.491  19.162  12.040  0.00  0.00      P1   C
ATOM    338  O   ASP A  21      29.367  19.523  12.441  0.00  0.00      P1   O
...
```

Note

Contrary to the hybrid36 standard of PDB files, ORCA writes non-standard pdb files as:

- atoms 1-99,999 in decimal numbers
- atoms 100,000 and beyond in hexadecimal numbers, with atom 100,000 corresponding to index 186a0.

This ensures a unique mapping of indices. If you want to select an atom with an index in hexadecimal space (index larger than 100,000), convert the hexadecimal number into decimals when choosing this index in the ORCA input file. Note also, that in the pdb file, counting starts from 1, while in ORCA counting starts from zero.

6.1.4 Active Atoms in Combination with Optimization, Frequency Calculation, MD

The systems of multiscale calculations can become quite large with tens and hundreds of thousands of atoms. In multiscale calculations the region of interest is often only a particular part of the system, and it is common practice to restrict the optimization to a small part of the system, which we call the active region = active part of the system. For info on how to define the active region see *QM and Active Region Definition*. Usually this active part consists of hundreds of atoms, and is defined as the QM region plus a layer around the QM region. The same definition holds for frequency calculations, in particular since after optimization non-active atoms are not at stationary points, and a frequency calculation would lead to artifacts in such a scenario. MD calculations on systems with hundreds of thousands of atoms are not problematic, but there are applications where a separation in active and non-active parts can be important (e.g. a solute in a solvent droplet, with the outer shell of the solvent frozen).

> **Note**
>
> If no active atoms are defined, the entire system is treated as active.

> **Note**
>
> The active region definitions also apply to MM calculations, but have to be provided via the `%qmmm` block.

Optimization in redundant internal coordinates

In ORCA's QM/MM geometry optimization only the positions of the active atoms are optimized. The forces on these active atoms are nevertheless influenced by the interactions with the non-active surrounding atoms. In order to get a smooth optimization convergence for quasi-Newton optimization algorithms in internal coordinates, it is necessary that the Hessian values between the active atoms and the directly surrounding non-active atoms are available. For that reason the active atoms are extended by a shell of surrounding non-active atoms which are also included in the geometry optimization, but whose positions are constrained, see Fig. 6.3. This shell of atoms can be automatically chosen by ORCA. There are three options available:

Table 6.2: Options to extend the active region.

Keyword	Options	Description
`EXTENDA(`	`DIST`	(Default) The parameter `Dist_AtomsAroundOpt` controls which non-active atoms are included in the extension shell, i.e. non-active atoms that have a distance of less than the sum of their VDW radii plus `Dist_AtomsAroundOpt` are included.
	`COV_`	All (non-active) atoms that are covalently bonded to active atoms are included.
	`NO`	No non-active atoms are included

The user can also provide the atoms for the extension shell manually. This will be discussed in section *Frequency Calculation*.

Fig. 6.3: Active and non-active atoms. Additionally shown is the extension shell, which consists of non-active atoms close in distance to the active atoms. The extension shell is used for optimization in internal coordinates and PHVA.

The set of active atoms is called the 'activeRegion', and the set of active atoms plus the surrounding non-active atoms is called 'activeRegionExt'. During geometry optimization the following trajectories are stored (which can be switched off):

Table 6.3: Files containing trajectories during optimization.

Filename	Description
basename_trj.xyz	Entire QM/MM system
basename.QMonly_trj.xyz	Only QM region
basename.activeRegion_trj.xyz	Only active atoms
basename.activeRegionExt_trj.xyz	Active atoms plus extension

The following files are stored containing the optimized structures - if requested:

Table 6.4: Files containing optimized structures.

Filename	Description
basename.pdb	Optimized QM/MM system in pdb file format
basename.xyz	Optimized QM/MM system
basename.QMonly.xyz	Only QM region
basename.activeRegion.xyz	Only active atoms
basename.activeRegionExt.xyz	Active atoms plus extension

Optimization with the Cartesian L-BFGS Minimizer

For very large active regions the quasi-Newton optimization in internal coordinates can become costly. It can be advantageous to use the `L-Opt` or `L-OptH` feature, see section *Geometry Optimizations using the L-BFGS optimizer*. In the context of multiscale simulations it can also be useful to treat parts of the system or full molecules of the system as rigid. E.g. you can optimize the hydrogen positions of the protein and water molecules, and at the same time relax non-standard molecules, for which no exact bonded parameters are available, as rigid bodies. For available options in that context see *Optimization Assuming a Rigid Body*. An example is shown here:

```
!QMMM L-OptH
%qmmm
 QMAtoms {22397:22423} end
 ORCAFFFilename "DNA_hexamer.ORCAFF.prms"
end
*pdbfile 0 1 DNA_ligand.pdb

%frag
 Definition
  1 {22307:22396} end # cofactor
  2 {22397:22423} end # ligand
 end
end

%geom
 RigidFrags {1 2} end # treat cofactor and ligand as individual rigid bodies
end
```

Frequency Calculation

If all atoms are active, a regular frequency calculation is carried out when requesting `!NumFreq`. If there are also non-active atoms in the QM/MM system, the *Partial Hessian Vibrational Analysis* (PHVA) is automatically selected. Here, the PHVA is carried out for the above defined `activeRegionExt`, where the extension shell atoms are automatically used as 'frozen' atoms. Note that the analytic Hessian is not available due to the missing analytic second derivatives for the MM calculation. Note that in a new calculation after an optimization it might happen that the new automatically generated extended active region is different compared to the previous region before optimization. This means that when using a previously computed Hessian (e.g. at the end of an optimization or a NEB-TS run) the Hessian does not fit to the new extended active region. ORCA tries to solve this problem by fetching the information on the extended region from the `.hess` file. If that does not work (e.g. if you distort the geometry after the Hessian calculation) you should manually provide the list of atoms of the extended active region. This is done via the following keyword:

```
%qmmm
 OptRegion_FixedAtoms {27 1288:1290 4400} end  # manually define the
                                               # activeRegionExt atoms.
end
```

Note that ORCA did print the necessary information in the output of the calculation in which the Hessian was computed:

```
Fixed atoms used in optimizer          ... 27 1288 1289 1290 4400
```

Nudged Elastic Band Calculations

NEB calculations (section *Nudged Elastic Band Method*) can be carried out in combination with the multiscale features in order to e.g. study enzyme catalysis. When automatically building the extension shell at the start of a Multiscale-NEB calculation, not only the coordinates of the main input structure ('reactant'), but also the atomic coordinates of the 'product' and, if available, of the 'transition state guess' are used to determine the union of the extension shell of the active region. For large systems it is advised to use the active region feature for the NEB calculation.

> **Note**
>
> The atomic positions of the non-active atoms of reactant and product and, if available, transition state guess, should be identical.

Molecular Dynamics

If there are active and non-active atoms in the multiscale system, only the active atoms are allowed to propagate in the MD run. If all atoms are active, all atoms are propagated. For more information on the output and trajectory options, see section *Regions*.

6.1.5 Boundary Region

This section is important for the QM/MM, QM/QM2 and QM/QM2/MM methods. For the latter method two boundary regions are present (between QM and QM2 as well as between QM2 and MM region), and both can go through covalent bonds. In the following we will only discuss the concept for the boundary between QM and MM, but the same holds true for the other two boundaries.

Link Atoms

ORCA automatically generates link atoms based on the information of the QM region and on the topology of the system (based on the `ORCAFF.prms` file). ORCA places link atoms on the bond between QM and MM atoms.

> **Important**
>
> When defining the QM, QM2 and MM regions, make sure that you only cut through single bonds (not aromatic, double, triple bonds, etc.).

The distance between QM atom and link atom is determined via a **bond length scaling factor** (in relation to the QM-MM bond length) that is computed as the ratio of the equilibrium bond length between QM and hydrogen atom (d_{QM-H}) and the equilibrium bond length between QM and MM atom (d_{QM-MM}).

For the **equilibrium bond lengths to hydrogen**, ORCA uses tabulated standard values for the most common atoms involved in boundary regions (C, N, O), which can be modified via keywords as defined further below. ORCA stores these values on-the-fly in a simple file (`basename.H_DIST.prms`). If necessary, the user can modify these values

atom-specific or add others to the file and provide this file as input to ORCA (see also paragraph *Get standard hydrogen bond lengths: getHDist*). For QM/QM2 and QM/QM2/MM methods the equilibrium bond lengths to hydrogen are explicitly calculated.

```
%qmmm
# standard equilibrium bond lengths with hydrogen can be modified
 Dist_C_HLA 1.09        # d0_C-H
 Dist_O_HLA 0.98        # d0_O-H
 Dist_N_HLA 0.99        # d0_N-H
# file can be provided which provides the used d0_X-H values specific to all atoms
 H_Dist_FileName "QM_H_dist.txt"
end
```

Bonded Interactions at the QM-MM Boundary

The MM bonded interactions within the QM region are neglected in the additive coupling scheme. However, at the boundary, it is advisable to use some specific bonded interactions which include QM atoms. By default ORCA neglects only those bonded interactions at the boundary, where only one MM atom is involved, i.e. all bonds of the type QM1-MM1, bends of the type QM2-QM1-MM1, and torsions of the type QM3-QM2-QM1-MM1. Other QM-MM mixed bonded interactions (with more than two MM atoms involved) are included. The user is allowed to include the described interactions, which is controlled via the following keywords:

```
%qmmm
 DeleteLADoubleCounting true     # Neglect bends (QM2-QM1-MM1) and torsions
                                 # (QM3-QM2-QM1-MM1). Default true.
 DeleteLABondDoubleCounting true # Neglect bonds (QM1-MM1)
end
```

Charge Alteration

If QM and MM atoms are connected via a bond (defined in the topology file), the charge of the close-by MM atom (and its neighboring atoms) has to be modified in order to prevent overpolarization of the electron density of LA and QM region. This charge alteration is automatically taken care of by ORCA in `%qmmm` block. ORCA provides the most popular schemes that can be used to prevent overpolarization.

Table 6.5: Options to modify the charge at the QM-MM boundary in order to prevent overpolarization.

Keyword	Options	Description
`ChargeAlter`	`CS`	(Default) Charge Shift - Shift the charge of the MM atom away to the MM2 atoms so that the overall charge is conserved
	`RCD`	Redistributed Charge and Dipole - Shift the charge of the MM atom so that the overall charge and dipole is conserved
	`Z0`	Keep charges as they are. This MM atom will probably lead to overpolarization
	`Z1`	Delete the charge on the MM1 atom (no overall charge conservation)
	`Z2`	Delete the charges on the MM1 atom and on its first (MM2) neighbors (no overall charge conservation)
	`Z3`	Delete the charges on the MM1 atom and on its first (MM2) and second (MM3) neighbors (no overall charge conservation)
`Scale_RCD`	`real`	For `RCD` scheme defines where on the bond between MM1 and MM2 the shifted charge is positioned. Default: `0.5`, i.e. midpoint.
`Scale_CS`	`real`	For `CS` scheme defines where on the bond between MM2 and MM1/MM3 the shifted charge is positioned. Default: `0.06` x bond.

6.1.6 Embedding Types

There are two embedding schemes available for treating the electrostatic interaction between high level and low level region. In the electrostatic embedding scheme the electrostatic interaction between QM and MM system is computed at the QM level. Thus, the charge distribution of the MM atoms (or other low level methods) can polarize the electron density of the QM region. The following embeding schemes are available:

Table 6.6: Options for embedding scheme to treat the electrostatic interaction between high and low level region.

Keyword	Options	Description
Embeddin	Electrostati	(Default) The electrostatic interaction between QM and MM system is computed at the QM level.
	Mechanical	The electrostatic interaction between QM and MM system is computed at the MM level.

Important

In QMMM the LJ interaction between QM and MM system is always computed at the MM level.

In the scheme of electrostatic embedding, the evaluation of the electrostatic potential generated by the MM part can be accelerated by using the FMM algorithm (described in reference [797]). This will speed up the building of the Fock Matrix. The default recommended setup can be called using the FMM-QMMM keyword directly in the keywords line. However, more details about the algorithm parameters and all input options can be found in (see also paragraph *Fast Multipole Method*)

```
! FMM-QMMM
```

It is recommended to use that option whenever the MM part is composed of more than 10,000 atoms (or point charges for ECM methods).

6.1.7 Additional Keywords

Here we list additional keywords and options that are accessible from within the `%qmmm` block and that are relevant to QM/MM calculations. Some of these keywords can also be accessed via the `%mm` block, see section *Available Keywords for the MM module*.

```
!QMMM
%qmmm

# The extended active region, activeRegionExt, contains the atoms surrounding the
# active atoms.
 ExtendActiveRegion distance  # rule to choose the atoms belonging
                              # to activeRegionExt.
                              # no - do not use activeRegionExt atoms
                              # cov_bonds - add only atoms bonded covalently to
                              # active atoms
                              # distance (default) - use a distance criterion (VDW
                              # distance plus Dist_AtomsAroundOpt)
 Dist_AtomsAroundOpt 0.0      # in Angstrom (Default 0). Meaning only freeze atoms
                              # which touch the active atoms by their VDW spheres.
                              # only needed for ExtendActiveRegion distance
 OptRegion_FixedAtoms {27 1288:1290 4400} end  # manually define the
                                               # activeRegionExt atoms.
                                               # Default: empty list.

# Printing options. All are true by default.
```

(continues on next page)

(continued from previous page)

```
 PrintLevel 1                # Can be 0 to 4, 0=nothing, 1=normal, ...
 PrintOptRegion true         # Additionally print xyz and trj for opt region
 PrintOptRegionExt false     # Additionally print xyz and trj for extended
                             # opt region
 PrintQMRegion true          # Additionally print xyz and trj for QM region
 PrintFullTRJ true           # Print trajectory of full system. Default true.
 PrintPDB true               # Additionally print pdb file for entire system, is
                             # updated every iteration for optimization

# Computing MM nonbonded interactions within non-active region.
 Do_NB_For_Fixed_Fixed true # Compute MM-MM nonbonded interaction also for
                             # non-active atom pairs. Default true.
# Treats all active water molecules that are treated on MM level as rigid bodies
#  in optimization and MD simulation, see section "Rigid Water".
 Rigid_MM_Water false        #  Default false.

end

*pdbfile 0 1 ubq.pdb  # structure input via pdb file, but also possible via
                      # xyz file
```

Table 6.7: Additional keywords in the `%qmmm` block, which are also accessible from the `%mm` block.

Keyword	Options	Description
ORCAFFFilen	"<filen	Filename of the ORCA Force Field prms file.
Do_NB_For_F	true/fa	Compute MM-MM nonbonded interaction also within non-active region, i.e. for non-active atom pairs. Default `true`. For `GOAT` and `Crystal-QMMM` default is `false`.
RIGID_MM_WA	true/fa	Optimization and MD of active MM waters. Default `false`.
ExtendActiv	distanc	Scheme to extend the active region to activeRegionExt. See also *here*. `distance`: Use a distance criterion (VDW distance plus `Dist_AtomsAroundOpt`).
	cov_bon	Add only atoms bonded covalently to active atoms.
	no	Do not add any atoms to activeRegionExt.
Dist_AtomsA	<real>	Distance to be added to VDW distance for `ExtendActiveRegion`. In Angstrom. Default `0`.
OptRegion_F	{ints}	List of atoms to define the extended active region. Default: empty list.
PrintOptReg	true/fa	Additionally print xyz and trj for opt region. Default `true`.
PrintOptReg	true/fa	Additionally print xyz and trj for extended opt region. Default `false`.
PrintPDB	true/fa	Additionally print pdb file for entire system, is updated every iteration for optimization. Default `false`.

6.2 Multiscale Simulations of molecular systems

ORCA features three methods suitable for molecular systems such as proteins, DNA, large molecules, explicit solvation, depending on their size and depending on the availability of force field parameters.

- *Additive QM/MM* is a pure additive QMMM method.
- *QM1/QM2* is a subtractive (2-layered) ONIOM method.
- *QM1/QM2/MM* is a 3-layered method, combining a subtractive method between QM1 and QM2, and an additive QMMM method for QM2/MM.

6.2.1 Additive QMMM

In the additive QMMM scheme force field parameters are used to describe the interactions in the MM region, as well as bonded interactions and LJ interactions between QM and MM atoms. Thus, force field parameters are required for the MM part of the calculation. ORCA currently supports the popular **CHARMM**, **AMBER**, and **Open Force Field** force fields, as discussed in *Molecular Mechanics Force Field in ORCA*, and provides tools to *convert* their force field parameter files into the ORCA force field file format.

Basic Usage

The minimum input necessary for an additive QM/MM calculation is shown in the following example. The force field filename is provided via the keyword `ORCAFFFilename`:

```
!QMMM
%qmmm
 QMAtoms {0 1 2 27 28} end
 ORCAFFFilename "UBQ.ORCAFF.prms"
end
```

> **Note**
>
> `ORCAFF.prms` files need to be provided for QM/MM, QM/QM2/MM and IONIC-CRYSTAL-QMMM calculations, but not for QM/QM2 and `MOL-CRYSTAL-QMMM`.

6.2.2 ONIOM Methods: General Overview

For the simulation of large systems with up to 10000 atoms, or for large QM regions in biomolecules, ORCA provides the QM/QM2 and QM/QM2/MM methods. The specifics of the two different methods are discussed further below. Here we are presenting the common concepts and keywords of both methods. For subtractive methods, we use a high level (QM) and a low level (QM2) of accuracy for different parts of the system. The advantages of this - in contrary to QM-MM methods - are as follows:

- QM2 methods are polarizable, the interaction with the high level region is more accurate.
- No MM parameters are needed for the atoms that are described at the QM2 level.

Available QM2 Methods

ORCA has several built in QM2 methods:

- Semiempirical methods (`AM1`, `PM3`)
- Tight-binding DFT (`XTB0`, `XTB1`, `XTB` (or `XTB2`))
- Composite methods (`HF-3c`, `PBEh-3c`, `r2SCAN-3c`)
- User-defined QM2 methods

The individual keywords for these methods are:

Table 6.8: Options for QM2 for QM/QM2 and QM/QM2/MM methods available via simple keywords.

QM/QM2	QM/QM2/MM
!QM/XTB	!QM/XTB/MM
!QM/XTB1	!QM/XTB1/MM
!QM/XTB0	!QM/XTB0/MM
!QM/GFN-FF	!QM/GFN-FF/MM
!QM/HF-3C	!QM/HF-3C/MM
!QM/PBEH-3C	!QM/PBEH-3C/MM
!QM/R2SCAN-3C	!QM/R2SCAN-3C/MM
!QM/PM3	!QM/PM3/MM
!QM/AM1	!QM/AM1/MM

Users can define their own low-level methods in the following way

```
!QM/QM2          or   !QM/QM2/MM
%qmmm
  QM2CUSTOMMETHOD "B3LYP"
  QM2CUSTOMBASIS "def2-SVP def2/J"
end
```

Alternatively, a custom QM2 method / basis set file can be provided:

```
!QM/QM2  or  !QM/QM2/MM
%qmmm
  QM2CustomFile "myQM2Method.txt" # File should be available
                                  # in working directory
end
```

The custom QM2 method file can contain any desired input, as e.g. the file myQM2Method.txt:

```
!cc-pVDZ HF TightSCF NOSOSCF KDIIS
%basis
  NewAuxJKGTO Mg "AutoAux" end
end
```

Note

Only add methods (including convergence settings) and basis sets for the QM2Custom options. Everything else (parallelization, memory, solvation, etc.) is taken care of by ORCA itself.

Electrostatic Interaction between high and low level

By default we are using electrostatic embedding, i.e. the high level system sees the atomic point charges of the low level (QM2) system. These point charges are derived from the full system low level (QM2) calculation. The following methods for determining these charges are available:

Table 6.9: Options for determining the atomic charges of the QM2 atoms.

Keyword	Option
Charge_Method	Hirshfeld (default)
	MBIS
	CHELPG
	Mulliken
	Loewdin (default if QM2 is AM1 or PM3)

The QM2 point charges can be scaled with the following keyword.

```
%qmmm
  Scale_QM2Charges_MMAtom 1. # default is 1.
end
```

Boundary Region

The boundary between high and low level part of the system can contain covalent bonds. For the detection and realistic treatment of these covalent bonds, a topology of the large QM2 system is generated at calculation start using the following keyword.

Table 6.10: Options for creating an initial topology for the full (QM/QM2) or medium (QM/QM2/MM) system.

Keyword	Option
AutoFF_QM2_Method	Simple_Eq (default)
	XTB
	XTB1
	XTB0
	GFNFF
	HF3C
	PBEH3C
	R2SCAN3C
	PM3
	AM1
	QM2

Note

If you use any of the XTB methods as AutoFF_QM2_Method make sure to have the otool_xtb binary in your ORCA PATH, see *Extended Tight-Binding: GFN0-xTB, GFN-xTB, GFN2-xTB*.

6.2.3 Subtractive QM/QM2 Method (ONIOM2)

The QM/QM2 method is a very convenient way of running multiscale calculations without the need to prepare any parameters. This method is a subtractive QM-QM method, in which we treat a part of interest on a higher level of accuracy, and the remainder of the system on lower level of accuracy. The implementation follows similar works as e.g. described in reference [798].

The method can be used in a similar way as a regular QM calculation. Let us have a look at the proton transfer in propionic acid, which can be modeled as follows:

```
!QM/XTB BP86 def2-TZVP def2/J
!Fast-NEB-TS NumFreq
!pal8
%qmmm
  QMAtoms {0:3} end
end
%neb
 product "propionicAcid_prod.xyz"
 preopt true
end
*xyz 0 1
H       -0.738352472      0.000000000     -5.836214279
O       -0.738352472     -0.587240971     -5.061536853
O       -0.738352472      1.434717404     -4.069730302
C       -0.738352472      0.227304724     -3.975502162
C       -0.738352472     -0.566448428     -2.687358498
H        0.133951528     -1.231202352     -2.710760176
H       -1.610656472     -1.231202352     -2.710760176
C       -0.738352472      0.318369069     -1.443687014
H       -0.738352472     -0.294739868     -0.538164669
H        0.142397528      0.965221387     -1.423275731
H       -1.619102472      0.965221387     -1.423275731
*
```

with the product structure file (propionicAcid_prod.xyz):

```
11
C3H6O2
H       -0.738352472      1.628728096     -5.020130139
O       -0.738352472     -0.587240971     -5.061536853
O       -0.738352472      1.434717404     -4.069730302
C       -0.738352472      0.227304724     -3.975502162
C       -0.738352472     -0.566448428     -2.687358498
H        0.133951528     -1.231202352     -2.710760176
H       -1.610656472     -1.231202352     -2.710760176
C       -0.738352472      0.318369069     -1.443687014
H       -0.738352472     -0.294739868     -0.538164669
H        0.142397528      0.965221387     -1.423275731
H       -1.619102472      0.965221387     -1.423275731
```

As can be seen from the input, the only difference to a regular calculation is the necessity to define the high level region via the `QMAtoms` or `QMCoreAtoms` keyword.

System charges and multiplicities

The two subsystems can have different (integer) charges and multiplicities. Defining the correct charges and multiplicities is important. The charge and multiplicity defined via the coordinate section defines the charge and multiplicity of the high level region (QMAtoms). The user still needs to define the charge and multiplicity of the total system (corresponding to the sum of the charge of the high level and low level parts, and corresponding to the overall multiplicity).

```
%qmmm
  QMAtoms {0:3} end # high level region
  Charge_Total  0   # charge of the full system. Default 0.
  Mult_Total    1   # multiplicity of the full system. Default 1.
end
*xyz 0 1            # charge and mult. of the high level region, i.e. atoms 0 to 3
```

Solvation

Implicit Solvation effects can be included in QM/QM2 calculations. On the one hand, for QM/XTB calculations, one can adopt the analytical linearized Poisson-Boltzmann (ALPB) solvation model, the domain decomposition COSMO (ddCOSMO), or the extended conductor-like polarizable continuum model (CPCM-X), and on the other hand, if no XTB is requested, ORCA uses the C-PCM. The user just needs to add the following tags in the ORCA input file, XTB for the QM2 region:

```
!QM/XTB ALPB(Water)

or

!QM/XTB DDCOSMO(Water)

or

!QM/XTB CPCMX(Water)
```

No XTB for the QM2 region:

```
!QM/HF-3c CPCM(Water)
```

If the ddCOSMO (XTB) or the C-PCM (non-XTB) are requested, there are two possible ONIOM/implicit-solvation methods [799].

- **C-PCM/B:** The effect of the solvent is, in the first place, included in the calculation for the large QM2 system. Once this calculation finishes, the solvation charges located on the surface of the cavity for the large system are used as point charges for the subsequent low-level and high-level calculations for the small system.
- **C-PCM/C:** The effect of the solvent is only included in the calculation for the large QM2 system.

The user can choose one scheme or the other via the tag "solv_scheme" in the "qmmm" block:

```
%qmmm
solv_scheme CPCM_B   # CPCM_B (default)
                     # CPCM_C
end
```

If the ALPB model or the CPCM-X are requested (within QM/XTB methods), the solvation effect is just included in the calculation for the large QM2 system (as one does for the C-PCM/C scheme).

6.2.4 QM/QM2/MM Method (ONIOM3)

The QM/QM2/MM method uses a combination of the subtractive scheme for the QM-QM2 part, and the additive scheme for the (QM-QM2) - (MM) interaction. It can be used if very large QM regions are required for biomolecules and explicitly solvated systems. The system is divided into a high level (QM), low level (QM2), and MM region (MM).

QM2 Atoms

QM2 atoms need to be defined for QM/QM2/MM calculations. They can be defined either directly

```
%qmmm
 QMAtoms {0:4} end           # list of QM atoms (counting starts from 0) or
 QM2Atoms {5:22} end         # list of QM2 atoms
end                          # an atom should not occur in both lists
*pdbfile 0 1 ubq.pdb
```

or via the occupancy column of a pdb file (see *QM and Active Region Definition*).

> **Note**
>
> QM2 atoms can be defined via 2 in the occupancy column. In this case `Use_QM2_InfoFromPDB` must be set to true.

System charges and multiplicities

The high and low level subsystems can have different (integer) charges and multiplicities. Defining the correct charges and multiplicities is important. The charge and multiplicity defined via the coordinate section defines the charge and multiplicity of the high level region (`QMAtoms`). The user still needs to define the charge and multiplicity of the medium system (corresponding to the sum of the charge of the high level and low level regions, and corresponding to the overall multiplicity of the combined high and low level region). The charge of the MM region is determined based on the MM parameters provided by the force field.

```
%qmmm
  QMAtoms {0:3} end # high level region
  Charge_Medium 0   # charge of the medium system. Default 0.
  Mult_Medium   1   # multiplicity of the medium system.  Default 0.
end
*xyz 0 1            # charge and mult. of the high level region, i.e. atoms 0 to 3
```

Basic Usage

An example for a QM/QM2/MM calculation looks as follows:

```
!QM/HF-3c/MM Opt B3LYP def2-TZVP def2/J NumFreq
%qmmm
 ORCAFFFilename "peptideChain.ORCAFF.prms"
 QMAtoms {16:33 68:82} end
 QM2Atoms {0:12 83:104} end
 ActiveAtoms { 0:38 65:120} end
 Charge_Medium 0
end
*pdbfile -1 1 peptideChain.pdb
```

6.3 CRYSTAL-QMMM

For the simulation of extended systems ORCA provides the CRYSTAL-QMMM features:

- *MOL-CRYSTAL-QMMM* for molecular crystals.
- *IONIC-CRYSTAL-QMMM* for semiconductors and insulators.

In this section we present the concepts and keywords that are common to both methods. ORCA is a molecular code and cannot carry out periodic calculations, but ORCA has been used for modeling selected properties for ionic solids using embedded cluster models already in the past (see e.g reference [599]). With ORCA 5 we provide the CRYSTAL-QMMM method, which simplifies setting up and running these types of embedded cluster calculations. The user needs to provide the structure with a (large enough) supercell representative for the crystal, and provide a list of QM atoms. The QM region should be located in the central part of the supercell. The QM atoms are embedded in the remainder of the supercell, the surrounding MM atoms, which are represented by atom-centered point charges and influence the QM calculations. How these charges are obtained, is outlined in the next paragraph.

6.3.1 Charge Convergence between QM and MM region

The core idea of the CRYSTAL-QMMM method is to reach charge convergence between the QM and the MM atoms. For this purpose, the MM charges are updated self-consistently with the atomic (CHELPG) charges of the QM atoms. This idea follows the work of reference [599] for the `IONIC-CRYSTAL-QMMM` and of reference [800] for the `MOL-CRYSTAL-QMMM`.

Table 6.11: Keywords for charge convergence settings for `!IONIC-CRYSTAL-QMMM` or `MOL-CRYSTAL-QMMM`

Keyword	Options	Description
`Conv_Charges`	`true/fa`	Use charge convergence scheme. Default `true`.
`Conv_Charges_M`	`<int>`	Maximum number of cycles for charge convergence scheme. Default `30` for MOL-CRYSTAL-QMMM. Default `50` for IONIC-CRYSTAL-QMMM.
`Conv_Charges_C`	`<real>`	Convergence threshold (default `0.01`) for maximum charge change for atom type between two subsequent charge convergence cycles.
`Scale_FormalCh`	`<real>`	MM atomic charges used in QM part of the CRYSTAL-QMMM calculation are scaled by this factor. Default `1.`.

During optimizations, the charge convergence scheme can be used (i) only at the very beginning, (ii) every N geometry steps, or (iii) for each geometry step, using the following keyword:

```
%geom
  ReConvCharge 1 # default is 1. Redo charge convergence scheme every N steps.
end
```

After charge convergence, the converged charges are stored in the file `basename.convCharges.ORCAFF.prms`. It can be used for a later calculation (with the same or different electronic structure settings) with the following keyword combination:

```
!IONIC-CRYSTAL-QMMM    or    !MOL-CRYSTAL-QMMM
%qmmm
  Conv_Charges false
  ORCAFFFilename "firstjob.convCharges.ORCAFF.prms"
end
```

6.3.2 MM-MM Interaction

Unlike with the other multiscale methods (QMMM, QM/QM2, QM/QM2/MM) the MM region is only treated as a perturbation to the QM region. The (bonded and nonbonded) MM-MM interaction is not computed - since it would not affect the QM-MM interaction - and thus the active region (optimizations, frequencies, …) in CRYSTAL-QMMM calculations should be restricted to atoms of the QM region.

6.3.3 Preparing CRYSTAL-QMMM Calculations

The input structures and input files for the CRYSTAL-QMMM calculations can be prepared with the *orca_crystalprep* module.

6.3.4 MOL-CRYSTAL-QMMM

Conceptually we use an additive QM/MM calculation, in which the QM region interacts with the MM region only via nonbonded interactions. Lennard-Jones interaction between QM and MM region is computed using van der Waals parameters from the UFF force field. The charge convergence treatment between QM and MM region starts with zero charges on the MM atoms, and is iterated until convergence of the QM atomic (CHELPG) and MM point charges is achieved.

The `MOL-CRYSTAL-QMMM` method expects as structural input a supercell that consists of repeating, identical molecular subunits, i.e. the atom ordering of the molecules should always be the same, and one molecular subunit should follow the next one. The minimum input necessary for a `MOL-CRYSTAL-QMMM` run looks as follows.

```
%qmmm
  NUnitCellAtoms 16       # provide the number of atoms per molecular subunit
  QMAtoms {160:175} end   # QM atoms, should be in central part of the supercell.
                          # Must be at least one complete molecular subunit.
end
```

> **Note**
>
> For molecular crystals it is assumed that the entire supercell consists of repeating units which each have zero charge. QM regions should consist of one or multiple of such charge-neutral units.

Extending the QM Region

ORCA can extend the QM region (which we call QM core region, defined by the original `QMAtoms` input) by one or multiple layers of molecular subunits using the `HFLayers` keyword. If the `HFLayers` keyword is used, the molecular subunits of the defined number of layers around the QM core region is added to the QM region. The layers of molecular subunits around the QM core region are detected via distance criteria.

```
%qmmm
  HFLayers 0                        # default 0
  HFLayerGTO "LANL2DZ"              # default. Use this basis set for the
                                    # HFLayer atoms.
  HFLayerECP "HayWadt"              # default. Use these ECPs for the HFLayer atoms.
  Conv_Charge_UseQMCoreOnly true    # only use the QM core region for the charge
                                    # convergence scheme
end
```

The HFLayer can be seen as a buffer region between the molecular subunit of interest (original QM atoms) and the surrounding subunits. The different possible reasons for `HFLayers` are as follows:

- For DLPNO calculations with `HFLayers`, the DLPNO multilevel feature is automatically switched on. The molecules of the `HFlayer` form an own fragment which is treated on HF level only, i.e. these molecules are not included in the Post-HF treatment.

- The `HFLayers` can also be used for non-DLPNO calculations. In such cases, the HFLayer molecules are in the same way included in the QM calculation as the other QM molecules. But their definition is a convenient way to choose a different basis set (and ECPs) for those molecules.
- It can be assumed that the QM charges computed for the QM core region are more realistic than the charges of the HFLayer atoms near the MM atomic charges. Thus, using only the QM atomic charges of the QM core region represent more realistic charges for the charge convergence scheme.
- If necessary, the atoms of the HFLayer can be defined by the user. Make sure that complete molecular subunits are used here.

```
%qmmm
  HFLayerAtoms {0:15} end
end
```

Note

The term `HFLayers` might be misleading. Strictly speaking, only for `MOL-CRYSTAL-QMMM` calculations with `DLPNO` methods (e.g. `DLPNO-CCSD(T)`, `DLPNO-MP2`, `DLPNO-B2PLYP`) the HF layer atoms are treated on HF level. For other methods (e.g. DFT) the HF layer atoms are treated with the same electronic structure method as the QM core region atoms.

Basic Usage

An example for a `MOL-CRYSTAL-QMMM` calculation looks as follows:

```
! MOL-CRYSTAL-QMMM
! PBE def2-SVP Opt NumFreq
%qmmm
 NUnitCellAtoms 13
 QMAtoms {221:233} end
 ActiveAtoms {221:233} end
end
*xyzfile 0 1 Ala_SC.xyz
```

6.3.5 IONIC-CRYSTAL-QMMM

Conceptually we use an embedded cluster calculation, in which the QM region interacts only with the MM atomic point charges of the surrounding. Unlike with `MOL-CRYSTAL-QMMM`, the Lennard-Jones interaction between QM and MM region is not computed. The charge convergence treatment between QM and MM region starts with the initial MM charges (the charges initially read from the ORCAFF.prms file) and is iterated until convergence of the QM atomic (CHELPG) and MM point charges is achieved.

Boundary Region

For ionic crystals a boundary region needs to be introduced between the QM region and the atomic point charges of the MM atoms in order to avoid spurious electron leakage from the clusters and to treat dangling bonds on the surface of the QM region. This boundary region consists of capped effective core potentials (cECPs). These cECPs are placed on the position of the MM atoms that are directly adjacent to the QM region. ORCA analyzes the connectivity of the atoms of the supercell and can thus detect the layers of atoms around the QM region, where the first layer consists of the atoms directly bonded to the QM atoms, the second layer consists of the atoms directly bonded to the atoms of the first layer and so on. The charges of these cECPs are determined in the same way as the MM atomic charges during the charge convergence scheme.

```
%qmmm
  ECPLayerECP "SDD"               # cECPs used for the boundary region
  ECPLayers 3                     # Number of cECP layers around the QM region.
                                  # Default is 3.
  Scale_FormalCharge_ECPAtom 1.   # default is 1. Charges on cECPs are scaled by
                                  # this factor
end
```

Extending the QM Region

ORCA can extend the QM region (which we call QM core region, defined by the original QMAtoms input) by one or multiple layers of atoms using the `HFLayers` keyword. If the `HFLayers` keyword is used, the atoms of the defined number of layers around the QM core region is added to the QM region. The layers of atoms around the QM core region are detected in the same way as defined above for the ECPLayers.

```
%qmmm
  HFLayers 0                        # default 0
  HFLayerGTO "LANL2DZ"              # default. Use this basis set for the
                                    # HFLayer atoms
  HFLayerECP "HayWadt"              # default. Use these ECPs for the HFLayer atoms.
  Conv_Charge_UseQMCoreOnly true # only use the QM core region for the charge
                                    # convergence scheme
end
```

The HFLayer can be seen as a buffer region between the actual atoms of interest (original QM atoms) and the surrounding cECP and MM point charge environment.The different possible reasons for `HFLayers` are as follows:

- For DLPNO calculations with `HFLayers`, the DLPNO multilevel feature is automatically switched on. The atoms of the HFLayer form an own fragment which is treated at HF level only, i.e. these atoms are not included in the Post-HF treatment.
- It can be assumed that the QM charges computed for the QM core region are more realistic than the charges of the HFLayer atoms near the cECPs and MM atomic charges, in particular for highly charged systems. Thus, using only the QM atomic charges of the QM core region represent more realistic charges for the charge convergence scheme.
- If necessary, the atoms of the HFLayer can be defined by the user:

```
%qmmm
  HFLayerAtoms {29:35} end
end
```

- The charge of the HFLayer is automatically computed based on the formal charges of its atoms. If necessary, the HFLayer charge can be provided by the user.

```
%qmmm
  Charge_HFLayer 10 # by default it is computed automatically based on the formal
                    # charges from the ORCAFF.prms file
end
```

Note

The term `HFLayers` might be misleading. Strictly speaking, only for `IONIC-CRYSTAL-QMMM` calculations with DLPNO methods (e.g. `DLPNO-CCSD(T)`, `DLPNO-MP2`, `DLPNO-B2PLYP`) the HF layer atoms are treated on HF level. For other methods (e.g. DFT) the HF layer atoms are treated with the same electronic structure method as the QM core region atoms.

Net Charge of the Entire Supercell

For ionic crystals, the QM region (as well as the entire supercell) can consist of an unequal number of atoms of each atom type. As a result of that, the QM region may have non-zero net charge. Consequently, when assigning the atomic point charges to the MM atoms during the charge convergence scheme, the sum of the charge of the QM region, of the ECP layer, and of the atomic charges of the MM atoms, can deviate from the ideal charge of the supercell. In order to enforce the ideal charge of the supercell, the total charge of the entire system is enforced by equally shifting the charges of all MM (including boundary) atoms.

```
%qmmm
  Charge_Total  0         # default 0. Total charge of the supercell
  EnforceTotalCharge true # enforce the total charge by shifting MM charges
end
```

The charge shifting procedure can be restricted to the MM atoms on the outer parts of the supercell by defining the number of `OuterPCLayers`. If `OuterPCLayers` is set to 1, only the atoms of the most outer layer of the supercell (recognized based on the connectivity of the atoms) are included in the charge shifting procedure. If `OuterPCLayers` is larger than 2, the atoms connected to the most outer layer are additionally included in the charge shifting procedure, etc.

```
%qmmm
  OuterPCLayers 0         # default 0, i.e. charge shifting for all MM atoms
end
```

Note

The charge of the QM core region can be chosen to be assigned automatically. This overwrites the charge provided in the xyz or pdb section.

```
%qmmm
  AutoDetectQCCharge false # default is false
end
```

Example Input

A minimal example for an `IONIC-CRYSTAL-QMMM` calculation looks as follows:

```
! IONIC-CRYSTAL-QMMM
! NMR
! PBE pcSseg-2 def2/JK def2-svp/C
%qmmm
  QMAtoms {0:6} end
  ORCAFFFilename "nacl6.ORCAFF.prms"
  CHARGE_TOTAL    0
end
*xyzfile -5 1 nacl6.xyz
```

Different QM and MM regions Stored in the PDB file

The stored PDB file contains the following entries in its occupancy column.

Table 6.12: Definition of entries stored in the pdb file for Ionic-Crystal-QMMM calculations.

Value	Description
0	MM atoms mimicked by surrounding point charges
1	QM core region atoms
2	HFLayer atoms
3	cECPs
4	OuterPCLayer atoms (subset of MM atoms) if defined, are the only atoms that are used in the charge shift procedure for enforcing the total charge

6.4 QM/MM via Interfaces to ORCA

ORCA is easy to interface as a QM engine in pretty much any QM/MM environment, as it will accept a set of point charges from an external file (see section *Inclusion of Point Charges*) and it will return, in a transparent format, all the required information for computing energies and gradients to the driving program. In our research group we have experience with four different QM/MM environments: ChemShell, Gromacs, pDynamo and NAMD. Each of the interfaces is described in the following section. It is beyond the scope of the manual to be exhaustive, and only minimal working examples are going to be presented. These will cover mainly the technical aspects with respect to the QM part of the QM/MM calculation. For the initial preparation of the system the user is referred to the documentation of the driving program.

6.4.1 ORCA and Gromacs

In cooperation with the developers of Gromacs software package we developed an interface to ORCA. The interface is available starting with Gromacs version 4.0.4 up to version 4.5.5.

As mentioned above, the initial setup has to be done with the Gromacs. In the QM/MM calculation Gromacs will call ORCA to get the energy and the gradients of the QM atoms. The interface can be used to perform all job types allowed by the Gromacs software package, e.g. optimizations, molecular dynamics, free energy. In addition, for geometry optimizations we have implemented a microiterative scheme that can be requested by setting the keyword `bOpt = yes` in the Gromacs `.mdp` file. This will cause ORCA to perform a QM geometry optimization every time it is called by Gromacs. During this optimization ORCA will also compute the Lennard-Jones interaction between the QM and MM atoms, and freeze the boundary atoms. This microiterative scheme can also be used to perform a transition state optimization. If you are looking for a transition state and have a good initial guess for the structure, you can carry out an optimization of the MM system and at the same time perform a transition state optimization of the QM system with ORCA via the microiterative scheme. Since it is expensive to calculate the Hessian for each microiterative step, the user can tell ORCA to use the (updated) Hessian matrix of the previous step via `read_temp_Hess` in the ORCA input.

In order to allow full flexibility to the user, the information for the QM run are provided in a separate plain text file, called `GromacsBasename.ORCAINFO`. When Gromacs writes the input for the ORCA calculation, it will merge this file with the information on the coordinates, point charges, Lennard-Jones coefficients and type of calculation (`EnGrad`, `Opt`, `TSOpt`).

Below is a typical example of an input file created by Gromacs, where for each Gromacs optimization step, a full optimization of the QM-part will be performed by ORCA, instead of just doing the energy and gradient calculation.

```
# Optimization step in the Lennard-Jones and point charges field of the MM atoms
! QMMMOpt

# file containing the Lennard Jones coefficients for the Lennard-Jones interaction
%LJCoefficients "temp.LJ"
```

(continues on next page)

(continued from previous page)

```
# file containing the point charges for the electrostatic interaction
%pointcharges "temp.pc"

%geom
    # calculate the exact Hessian before the first optimization step
    Calc_Hess true
    # in case of a TS optimization the updated Hessian of the previous
    # TS optimization run is read instead of calculating a new one
    read_temp_Hess true
 end
```

Note

- If you are using `bOpt` or `bTS` you have to take care of additional terms over the boundary. Either you can zero out the dihedral terms of the Q2-Q1-M1-M2 configurations, or you can fix the respective Q2 atoms.
- If you want to use the ORCA constraints, you have to also put them in the Gromacs part of the calculation.
- If there are no bonds between the QM and MM systems, the `bOpt` optimization of the QM system might have convergence problems. This is the case if the step size is too large and thus QM atoms come too close to MM atoms. The convergence problems can be circumvented by adding extra coordinates and Hessian diagonal values for Cartesian coordinates of selected QM atoms to the set of internal coordinates. This should be done for only few atoms that are in the boundary region.

```
%geom
    # add the Cartesian position of atoms 2 and 4 to the
    # set of internal coordinates with a diagonal Hessian value of 0.1
    Hess_Internal
    {C 2 D 0.1}
    {C 4 D 0.1}
    end
end
```

6.4.2 ORCA and pDynamo

The interface with the pDynamo library is briefly documented here. It uses the same plain text files to exchange information between the pDynamo library and ORCA. In order to have simiar functionality as presented above, we have extended the interface to generate more complex ORCA input files by accepting verbatim blocks of text. We have also implemented in pDynamo the capability of writing files containing Lennard-Jones parameters.

A simple example which calculates the QM/MM energy for a small designed peptide in which the side chain of one amino acid is treated QM is ilustrated below. For this example, a complete geometry optimization is going to be performed during the ORCA call.

```
import os
import pCore
import pBabel
import pMolecule
import pMoleculeScripts

# Read the initial coordinates from the .pdb file.
system = pMoleculeScripts.PDBFile_ToSystem(
        "1UAO.pdb", modelNumber=1, useComponentLibrary=True)

# Instantiate the required models.
mmmodel = pMolecule.MMModelOPLS("protein")
nbmodel = pMolecule.NBModelORCA()
qcmodel = pMolecule.QCModelORCA(
```

(continues on next page)

(continued from previous page)

```
        command=os.getenv("ORCA_COMMAND"),
        deleteJobFiles=False, header="bp86 def2-svp qmmmopt/pdynamo",
        job="chignolin", run=True)

# Assign the models to the system.
system.DefineMMModel(mmmodel)
system.DefineQCModel(
        qcmodel, qcSelection=pCore.Selection([35, 36, 37, 34, 40, 41]))
system.DefineNBModel(nbmodel)
system.electronicState = pMolecule.ElectronicState(
        charge=-1, multiplicity=1)

# Print a summary and calculate the energy.
system.Summary()
system.Energy()
```

After the execution of the above Python program, a series of files are going to be created `chignolin.inp`, `chignolin.pc`, `chignolin.lj` and ORCA is going to be called. The generated ORCA input file is listed below.

```
! bp86 def2-svp qmmmopt/pdynamo
% geom
    constraints
        {C 0 C}
        {C 1 C}
        end
    end

% pointcharges "chignolin.pc"
% ljcoefficients "chignolin.lj"
* xyz -1 1
H                  -1.0637532468        1.1350324675        2.4244220779
C                  -0.5230000000        0.6870000000        3.2490000000
C                   0.4180000000        1.7240000000        3.8660000000
O                  -0.0690000000        2.7620000000        4.2830000000
O                   1.6090000000        1.4630000000        3.9110000000
H                  -1.2240000000        0.3460000000        3.9970000000
H                   0.0550000000       -0.1510000000        2.8890000000
*
```

There are few points that have to be raised here. Because the keyword `qmmm/pdynamo` was specified in the `header` variable, the pDynamo library will automatically add the `constraint` block in the ORCA input, which will freeze the link atoms and the QM atoms to which they are bound. It will also generate the `chignolin.lj` file containing all the Lennard-Jones parameters. The important parts of this file, which is somewhat different than the one generated by Gromacs, are listed next.

```
# number of atoms  combination rule
138 0
#      x            y            z          sigma       epsilon    id
   -6.778000    -1.424000     4.200000     3.250000     0.711280   -1
   -6.878000    -0.708000     2.896000     3.500000     0.276144   -1
   -5.557000    -0.840000     2.138000     3.750000     0.439320   -1
...
    0.433000     0.826000     0.502000     2.960000     0.878640   -1
   -0.523000     0.687000     3.249000     3.500000     0.276144    1
    0.418000     1.724000     3.866000     3.750000     0.439320    2
   -0.069000     2.762000     4.283000     2.960000     0.878640    3
    1.609000     1.463000     3.911000     2.960000     0.878640    4
   -2.259000    -0.588000     1.846000     0.000000     0.000000   -1
   -1.795000     2.207000     2.427000     2.500000     0.125520   -1
   -1.224000     0.346000     3.997000     2.500000     0.125520    5
```

(continues on next page)

(continued from previous page)

```
    0.055000    -0.151000     2.889000     2.500000     0.125520    6
   -0.311000     2.922000     0.557000     3.250000     0.711280   -1
...
   -1.387000    -2.946000     5.106000     2.500000     0.125520   -1
# number of special pairs
22
#        atom1        atom2    factor
          34           32     0.000000
          35           39     0.500000
          40           31     0.000000
          41           30     0.500000
          41           32     0.500000
          36           31     0.500000
          40           32     0.500000
          40           39     0.500000
          34           31     0.000000
          35           30     0.500000
          34           11     0.500000
          34           38     0.500000
          41           31     0.000000
          37           31     0.500000
          34           33     0.500000
          34           39     0.000000
          40           30     0.500000
          41           39     0.500000
          34           30     0.000000
          35           31     0.000000
          34           42     0.500000
          35           32     0.500000
```

The second number on the first line refers to the type of combination rule used to calculate the Lennard-Jones interaction. It is 0 if a geometric average is used (OPLS force field), or 1 for the Lorentz-Berthelot rules (AMBER force field). The `id` on the last column is -1 for MM atoms and is equal to the atom number for the QM atoms. In this case the hydrogen link atom is atom 0. The last block of the file is composed of atom pairs and a special factor by which their Lennard-Jones interaction is scaled. In general this factor is equal to 1, but for atoms one or two bonds apart is zero, while for atoms three bonds apart depends on the type of force field, and in this case is 0.5.

After successful completion of the ORCA optimization run, the information will be relayed back the pDynamo library, which will report the total QM/MM energy of the system. At this point the type QM/MM of calculation is limited only by the capabilities of the pDynamo library, which are quite extensive.

6.4.3 ORCA and NAMD

Since version 2.12, NAMD is able to perform hybrid QM/MM calculations. A more detailed explanation of all available key words, setting up the calculation and information on tutorials and on the upcoming graphic interface to VMD are available on the NAMD website.

Similar to other calculations with NAMD, the QM/MM is using a pdb file to control the active regions. An example is shown below, where the sidechain of a histidine protonated at Nϵ is chosen to be the QM region. Either the occupancy column or the b-factor column of the file are used to indicate which atom are included in a QM area and which are treated by the forcefield. In the other column, atoms which are connecting the QM area and the MM part are indicated similarly. To clarify which column is used for which purpose, the keywords `qmColumn` and `qmBondColumn` have to be defined in the NAMD input.

```
...
ATOM   1737  CA  HSE P 117      14.762  47.946  31.597  1.00  0.00      PROT C
ATOM   1738  HA  HSE P 117      14.751  47.579  32.616  0.00  0.00      PROT H
ATOM   1739  CB  HSE P 117      14.129  49.300  31.501  1.00  1.00      PROT C
ATOM   1740  HB1 HSE P 117      14.407  49.738  30.518  0.00  1.00      PROT H
```

(continues on next page)

(continued from previous page)

```
ATOM   1741  HB2 HSE P 117      13.024  49.194  31.509  0.00  1.00      PROT H
ATOM   1742  ND1 HSE P 117      13.899  51.381  32.779  0.00  1.00      PROT N
ATOM   1743  CG  HSE P 117      14.572  50.261  32.582  0.00  1.00      PROT C
ATOM   1744  CE1 HSE P 117      14.615  52.043  33.669  0.00  1.00      PROT C
ATOM   1745  HE1 HSE P 117      14.356  53.029  34.064  0.00  1.00      PROT H
ATOM   1746  NE2 HSE P 117      15.678  51.318  33.982  0.00  1.00      PROT N
ATOM   1747  HE2 HSE P 117      16.369  51.641  34.627  0.00  1.00      PROT H
ATOM   1748  CD2 HSE P 117      15.706  50.183  33.335  0.00  1.00      PROT C
ATOM   1749  HD2 HSE P 117      16.451  49.401  33.388  0.00  1.00      PROT H
ATOM   1750  C   HSE P 117      13.916  47.000  30.775  0.00  0.00      PROT C
ATOM   1751  O   HSE P 117      12.965  46.452  31.334  0.00  0.00      PROT O
...
```

NOTES:

- If one wants to include more than one QM region, integers bigger than 1 can be used to define the different regions.
- Charge groups cannot be split when selecting QM and MM region. The reason is that non-integer partial charges may occur if a charge group is split. Since the QM partial charges are updated in every QM iteration, this may lead to a change in the total charge of the system over the course of the MD simulation.
- The occupancy and b-factor columns are used for several declarations in NAMD. If two of these come together in one simulation, the keyword `qmParamPDB` is used to define which pdb file contains the information about QM atoms and bonds.
- To simplify the selection of QM atoms and writing the pdb file a set of scripts is planned to be included in future releases of NAMD.

To run the calculation, the keyword `qmForces on` must be set. To select ORCA `qmSoftware "orca"` must be specified and the path to the executables must be given to `qmExecPath`, as well as a directory where the calculation is carried out (`qmBaseDir`). To pass the method and specifications from NAMD to ORCA `qmConfigLine` is used. These lines will be copied to the beginning of the input file and can contain both simple input as well as block input. To ensure the calculation of the gradient, the `engrad` keyword should be used.

The geometry of the QM region including the selected links as well as the MM point charges are copied to the ORCA inputfile automatically. Multiplicity and charge can be defined using `qmMult` and `qmCharge`, although the latter can be determined automatically by NAMD using the MM parameters. It should be noted at this point that NAMD is capable to handle more than one QM region per QM/MM calculation. Therefore for each region, charge and multiplicity are expected. In the case of only one QM region, the input looks like the following:

```
qmMult          "1 1"
qmCharge        "1 0"
```

Currently, two charge modes are available: Mulliken and CHELPG. They have to be specified in the NAMD input using `QMChargeMode` and in the `qmConfigLine`, respectively. Different embedding schemes, point charge schemes and switching functions are available, which will be not further discussed here. Another useful tool worth mentioning is the possibility to call secondary executables before the first or after each QM software execution using `QMPrepProc` or `QMSecProc`, respectively. Both are called with the complete path and name to the QM input file, allowing e.g. storage of values during an QM/MM-MD.

It is strongly emphasized that at this points both programs are constantly developed further. For the latest information, either the ORCA forum or the NAMD website should be consulted.

6.5 Fast Multipole Method

The Fast Multipole Method (FMM) algorithm was proposed in the 1980s[797], to reduce the computation time for two-centre interactions in large systems, by moving from a quadratic relationship (O(N^2)) to a quasi-linear relationship (O($Nlog(N)$)) of the computation time with the number N of particles (atoms, point charges, etc.). This algorithm is particularly useful for long-range interactions that are difficult to ignore, such as the Coulombic interaction (1/r) which is fundamental to any system but grows quadratically with the number of centers.

The FMM is used in ORCA for accelerating QMMM calculations in the scope of electrostatic embedding: FMM-QMMM (cf. *Embedding Types*).

6.5.1 The Octree hierarchy

There exists a lot of detailed and pedagogical literature on the subject (we recommend for instance [801] and [802]). In the following we will only describe the main parameters.

The idea of the algorithm is to divide space to handle differently short- and long-range interactions. The latter will be approximated (cf. *Fast Multipole Method*). To do so, the whole system is placed in a cubic box which is iteratively divided in 8 children boxes (forming levels L: L $= 0 \rightarrow$ L $=$ L_{max}) so as to form a structure called an Octree (see Fig. Fig. 6.4).

Fig. 6.4: Division of the system into boxes, schematic representation in 2D.

At the deepest level, L $=$ L_{max}, for every box, we can define a layer of nearest neighbours boxes (NN) which will be responsible for the Near-Field (NF, i.e.short-range). The electrostatic potential due to the rest of the boxes is define as the Far-Field (FF, i.e. long-range).

$$\forall B \ \ , \ \ \mathbf{V}_B = \mathbf{V}_B^{\mathbf{NF}} + \mathbf{V}_B^{\mathbf{FF}}$$

As visible on Figure Fig. 6.4, one can divide the FF area between a Local Far Field (LFF, green) area and a Remote Far Field area (RFF, grey), so that a recursive scheme between levels appears:

$$\begin{aligned} \forall B \ \ \mathbf{V}_B^{\mathbf{FF}} &= \mathbf{V}_B^{\mathbf{LFF}} + \mathbf{V}_B^{\mathbf{RFF}} \\ &= \mathbf{V}_B^{\mathbf{LFF}} + \mathbf{W}^T_{Parent(B)\rightarrow B}\mathbf{V}^{\mathbf{FF}}_{Parent(B)} \end{aligned}$$

The LFF boxes are the boxes in the NF of the parent but not in the NF of the children, that represents a maximum of 189 boxes. The RFF potential is due to boxes which actually represent the FF area of the parent box of B. This

means that one needs to calculate only the LFF term at every level, the rest of the potential will be inherited from the parent box.

The number of levels, L_{max}, can be setup in the input directly (FMMQMMM_Levels), or will be deduced from the provided box dimension (FMMQMMM_BoxDimInp, dimension of the box at L_{max}). The second option (default) is recommended, with a box dimension around 9.0 Bohr (FMMQMMMM_BoxDimInp 9.0). If FMMQMMM_DoBoxDimOpt option is turned to TRUE (default), the box dimension will be reduced as much as possible while keeping the same value of L_{max}, this to ensure the algorithm works optimally regarding both accuracy and efficiency.

```
%method
  DOFMMQMMM              True  #turn ON the use of the FMM
  FMMQMMM_BoxDimInp       9.0  #Higher boundary to set up the dimension (in Bohr!)
  FMMQMMM_DoBoxDimOpt    True  #Optimize the box dimension
end
```

The higher L_{max}, the bigger the number of boxes: at $L = 2$ one has 64 boxes only but for $L = 6$ the code generates more than 200,000 boxes (8^6). For $L_{max} \geq 7$, more than 2 millions boxes are generated, this can start leading to memory issues, so that the user should ensure enough memory is available (cf. *Global Memory Use*).

6.5.2 Approximation of the Far Field interactions

In a space of origin O(0,0,0), let P ($\mathbf{r}_P$) be the center of a charge distribution (e.g. a Gaussian overlap or a distribution of point charges). Let's consider a point A in the vicinity of P, such that A = (q_A; $\mathbf{r}_A$), with q_A the charge of a point charge or $q_A = -1$ if we consider a Gaussian overlap. In that case, the charge would indeed be the one of the electron situated at a position $\mathbf{r}_{AP} = \mathbf{r}_A - \mathbf{r}_P$ from the center of the Gaussian overlap. Similarly, let Q ($\mathbf{r}_Q$) be the center of a charge distribution and B, a point in the vicinity of Q, such that B = (q_B; $\mathbf{r}_B$). We note $\mathbf{r}_{QP} = \mathbf{r}_Q - \mathbf{r}_P$.

The Coulomb interaction between A and B can be expressed as:

$$U_{AB} = f\Big(\frac{1}{|\mathbf{r}_a - \mathbf{r}_b|}\Big),$$

The name of the algorithm comes from the way the FF interactions are evaluated. It is done through a mutlipole expansion of the $\frac{1}{|\mathbf{r}_a - \mathbf{r}_b|}$ term, leading to the following expression:

$$\frac{1}{|\mathbf{r}_A - \mathbf{r}_B|} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{j=0}^{\infty} \sum_{k=-j}^{j} (-1)^j R_{l,m}(\mathbf{r}_{AP}) I_{l+j,m+k}(\mathbf{r}_{QP}) R_{j,k}(\mathbf{r}_{BQ}),$$

This series converges for well separated centers only, that is why a NF layer is required. $R_{l,m}$ and $I_{l,m}$ are regular and irregular solid scaled harmonics [803]:

$$R_{l,m}(\mathbf{r}) = \frac{1}{\sqrt{(l-m)!(l+m)!}} \mathbf{r}^l \sqrt{\frac{4\pi}{2l+1}} Y_{l,m}(\mathbf{r})$$
$$I_{l,m}(\mathbf{r}) = \sqrt{(l-m)!(l+m)!} \frac{1}{\mathbf{r}^{l+1}} \sqrt{\frac{4\pi}{2l+1}} Y_{l,m}(\mathbf{r})$$

$Y_{l,m}$ are spherical harmonics of degree l and order m. For details on the derivation of this equation see [801].

We can now introduce the multipole expansion centered in P ($\mathbf{r}_P$), of a charge distribution $\mathbf{Q}(\mathbf{r}, P)$, as:

$$\mathbf{Q}(\mathbf{r}, P) = f\Big(\sum_{l=0}^{\infty} \sum_{m=-l}^{l} R_{l,m}(\mathbf{r} - \mathbf{r}_P)\Big)$$

In practise:

$$\mathbf{Q}(\mathbf{r}, P) \approx f\Big(\sum_{l=0}^{MAM} \sum_{m=-l}^{l} R_{l,m}(\mathbf{r} - \mathbf{r}_P)\Big)$$

Q is a vector, with elements $Q_{l,m}$ defined by the pair of values (l,m). In the case of a point charge A (q$_A$,**r**) we have:

$$Q^A_{l,m}(\mathbf{r}, P) = q_A \times R_{l,m}(\mathbf{r} - \mathbf{r}_P),$$

while for a Gaussian overlap distribution $\langle\mu|\nu\rangle$ it becomes:

$$\begin{aligned} Q^{\mu\nu}_{l,m}(\mathbf{r}, P) &= \langle\mu| R_{l,m}(\mathbf{r} - \mathbf{r}_P) |\nu\rangle \\ &= \int \mu(\mathbf{r'})R_{l,m}(\mathbf{r} - \mathbf{r}_P)\nu(\mathbf{r'})d\mathbf{r'} \end{aligned}$$

So that we can approximate the electrostatic interaction between a Gaussian overlap $\langle\mu|\nu\rangle$ with center in $\mathbf{r}_B$ and a point charge A (q$_A$,$\mathbf{r}_A$) in the FF, through the following expansion:

$$\frac{\int q_A\mu(\mathbf{r})\nu(\mathbf{r})d\mathbf{r}}{|\mathbf{r}_A - \mathbf{r}_B|} \approx \mathbf{Q}^{\mu\nu}(\mathbf{r}_b, P')\mathbf{I}(\mathbf{r}_{PP'})\mathbf{Q}^A(\mathbf{r}_A, P),$$

with P and P' the respective center of the multipole expansions, and I the interaction matrix.

The truncation parameter of the expansion, MAM, is the Maximum Angular Momentum of the underlying solid scaled harmonics. In our setup, it is the same value for the expansion of the point charges in the embedding or the Gaussian overlaps in the QM part. It can be setup through the keyword FMMQMMM_MAM. We recommend FMMQMMM_MAM = 20 to ensure a good accuracy, in the order of 0.0001-0,0001 Ha. With a lower value of MAM=15, one can usually expect a mHa (0.63 kcal.mol^{-1}) precision. However it is system dependent, so that we recommend you to start with MAM=20 and see how it behaves if you decrease it to a value of 15. The available values for MAM are in the range 1-25.

```
%method
  DOFMMQMMM              True  #turn ON the use of the FMM
  FMMQMMM_MAM              20   #Default Maximum Angular Momentum
end
```

6.5.3 The "Very" Fast Multipole Method (VFMM)

Due to the recursive scheme between levels, the FF felt in the center of a box is built by only calculating interactions with the LFF boxes at every level. The evaluation of V^{LFF} becomes the bottleneck of the algorithm. To accelerate it, an option has been implemented in which different MAM truncation parameters will be used for all the (>189) boxes in the LFF area. This option FMMQMMMM_DoVFMM is turned ON by default, but should be switched OFF whenever the MAM is smaller than 15, that would impact to much the accuracy without improving much the efficiency.

```
%method
  DOFMMQMMM              True  #turn ON the use of the FMM
  FMMQMMM_DoVFMM         True  #Use the Very Fast Multipole Method
end
```

6.5.4 Recommended input

Whenever there is an electrostatic embedding, for systems containing more than 10,000 point charges (ECM) or MM atoms (QMMM), it is recommended to turn on the FMM in order to accelerate the calculation of the electrostatic potential. However, when using heavy parallelization with more than 24 processors, the impact of enabling the Fast Multipole Method (FMM) may be negligible for small embedding size.

Caution

FMM-QMMM does not replace the QMMM keyword.

The two elements one can play on are the truncation parameter of the expansion, MAM, and the box dimension at the deepest level (L_{max}). Recommended/default parameters can be called in the keyword line directly using FMM-QMMM or by setting the parameters accordingly in the %method block:

```
#keyword line
! FMM-QMMM

#OR through method block

%method
  DOFMMQMMM             True  #turn ON the use of the FMM
  USEFMMQMMM            True  #turn ON the use of the FMM
  ###parameters:
  FMMQMMM_DoVFMM        True  #Use the Very Fast Multipole Method
  FMMQMMM_MAM             20  #Maximum Angular Momentum
  FMMQMMM_Levels           0  #Number of levels set to 0, will be set␣
↪automatically by box dim
  FMMQMMM_BoxDimInp      9.0  #Higher boundary to set up the dimension (in Bohr!)
  FMMQMMM_DoBoxDimOpt   True  #Optimize the box dimension
end
```

The parameters used by the algorithm are printed in the output by default, we recomend you to check them in your first calculations:

```
----------------------
SHARK INTEGRAL PACKAGE
----------------------

[...]

Use FMM for one center integrals with PCs

- - - - - - - FMM PARAMETERS- - - - - - -
num PCs              75894
VFMM                  USED
MAM                   20
Input box dim        9.000
Refined box dim      5.197
Tree depth/levels      6
- - - - - - - - - - - - - - - - - - - - -
```

This can be turned off by modifying the relative printing options.

```
%method
 FMMQMMM_Printing 1 # remove FMM PARMETERS printing, default value is  2
end
```

6.5.5 Some examples

Call the recommanded FMM parameters through the keyword line:

```
!QMMM wB97X-D3BJ RIJCOSX FMM-QMMM

%qmmm
 ORCAFFFilename "file.prms"
 Embedding Electrostatic
 ChargeAlteration CS
 QMAtoms {2016:2026} end
end
```

(continues on next page)

(continued from previous page)

```
*pdbfile 0 1 file.pdb
```

Change the MAM from 20 (value when one uses the keyword line) to 15, and use 5 levels:

```
!QMMM wB97X-D3BJ RIJCOSX

%method
  DOFMMQMMM             True  #turn ON the use of the FMM
  USEFMMQMMM            True  #turn ON the use of the FMM
  ###parameters:
  FMMQMMM_DoVFMM        True  #Use the Very Fast Multipole Method
  FMMQMMM_MAM             15  #Maximum Angular Momentum
  FMMQMMM_Levels           5  #number of levels
end

%qmmm
 ORCAFFFilename "file.prms"
 Embedding Electrostatic
 ChargeAlteration CS
 QMAtoms {2016:2026} end
end


*pdbfile 0 1 file.pdb
```

Specify the box dimension without optimizing it:

```
!QMMM wB97X-D3BJ RIJCOSX

%method
  DOFMMQMMM             True  #turn ON the use of the FMM
  USEFMMQMMM            True  #turn ON the use of the FMM
  ###parameters:
  FMMQMMM_DoVFMM        True  #Use the Very Fast Multipole Method
  FMMQMMM_MAM             15  #Maximum Angular Momentum
  FMMQMMM_Levels           0  #number of levels
  FMMQMMM_BoxDimInp      9.0  #Higher boundary to set up the dimension (in Bohr!)
  FMMQMMM_DoBoxDimOpt   False #Does not optimize the box dimension
end

%qmmm
 ORCAFFFilename "file.prms"
 Embedding Electrostatic
 ChargeAlteration CS
 QMAtoms {2016:2026} end
end


*pdbfile 0 1 file.pdb
```

Specifying the box dimension and ask to optimize it:

```
!QMMM wB97X-D3BJ RIJCOSX

%method
  DOFMMQMMM             True  #turn ON the use of the FMM
  USEFMMQMMM            True  #turn ON the use of the FMM
  ###parameters:
  FMMQMMM_DoVFMM        True  #Use the Very Fast Multipole Method
  FMMQMMM_MAM             15  #Maximum Angular Momentum
```

(continues on next page)

(continued from previous page)

```
  FMMQMMM_Levels           0  #number of levels
  FMMQMMM_BoxDimInp      9.0  #Higher boundary to set up the dimension (in Bohr!)
  FMMQMMM_DoBoxDimOpt   True  #Optimize the box dimension
end

%qmmm
 ORCAFFFilename "file.prms"
 Embedding Electrostatic
 ChargeAlteration CS
 QMAtoms {2016:2026} end
end


*pdbfile 0 1 file.pdb
```

CHAPTER

SEVEN

MOLECULAR DYNAMICS

7.1 *Ab initio* Molecular Dynamics

A few years ago, we included an *ab initio* molecular dynamics (AIMD) module into ORCA.[1] As a plethora of different electron structure methods with analytical gradients is already implemented, all these methods are now available also for MD simulations, offering a wide range of accuracy/performance trade-offs.

Despite the relatively short history inside of the ORCA package, the MD module has grown considerably over the last years. A few features found in other MD codes are still missing. In future releases, many more new features and methods will (hopefully) be added to this part of the program. We will always do our best to keep a strict backward compatibility, such that the sample inputs from this section will remain valid in all future releases.

For some more information as well as input examples for the ORCA MD module, please visit

https://brehm-research.de/orcamd

7.1.1 Changes in ORCA 5.0

- Added a `Metadynamics` module with many features and options:
  - Can perform one-dimensional and two-dimensional Metadynamics simulations [804] to explore free energy profiles along reaction coordinates, called collective variables ("Colvars").
  - `Colvars` can be distances *(including projections onto vectors and into planes)*, angles, dihedrals, and coordination numbers [805]. The latter allows, *e. g.*, to accurately compute pK_A values of weak acids [806, 807].
  - For all Colvars, groups of atoms *(e. g., centers of mass)* can be used instead of single atoms.
  - Metadynamics simulations can be easily restarted and split over multiple runs.
  - Ability to run well-tempered Metadynamics [808] for a smoothly converging free energy profile.
  - Ability to run extended Langrangian Metadynamics [805], where a virtual particle on the bias profile is coupled to the real system via a spring. The virtual particle can be thermostated.
- Added two modern and powerful `thermostats` *(both available as global and massive)*:
  - The widely used Nosé–Hoover chain thermostat (NHC) with high-order Yoshida integrator [809, 810]; allows for a very accurate sampling of the canonical ensemble.
  - The stochastical "Canonical Sampling through Velocity Rescaling" (CSVR) thermostat [811] which has become quite popular recently.
- Can define harmonic and Gaussian `restraints` for all Colvars *(distance, angle, dihedral, coordination number)*. This allows for umbrella sampling [812, 813], among other methods. Can also define one-sided restraints which act as lower or upper wall.

[1] Strictly speaking, these simulations are Born–Oppenheimer molecular dynamics simulations (BOMD), because they approximately solve the time-independent Schrödinger equation to compute gradients and then move the atoms according to these gradients.

- Can now print the instantaneous and average force on `constraints` and `restraints`; this allows for thermodynamic integration [813].
- The target value for `constraints` and `restraints` can now be a ramp, so that it can linearly change during the simulation.
- Can now keep the system's center of mass fixed during MD runs.
- Can now print population analyses, orbital energies, and .engrad files in every MD step if requested.

7.1.2 Changes in ORCA 4.2 (Aug 2019)

- Added a Cartesian minimization command to the MD module, based on L-BFGS and simulated annealing. Works for large systems ($> 10\,000$) atoms and also with constraints. Offers a flag to only optimize hydrogen atom positions *(for crystal structure refinement)*. See `Minimize` command.
- The MD module can now write trajectories in DCD file format *(in addition to the already implemented XYZ and PDB formats)*, see `Dump` command.
- The `thermostat` is now able to apply temperature ramps during simulation runs.
- Added more flexibility to region definition *(can now add/remove atoms to/from existing regions)*. Renamed the `Define_Region` command to `Manage_Region`.
- Added two new constraint types which keeps centers of mass fixed or keep complete groups of atoms rigid, see `Constraint` command.
- Ability to store the GBW file every n-th step during MD runs *(e.g. for plotting orbitals along the trajectory)*, see `Dump` command.
- Can now set limit for maximum displacement of any atom in a MD step, which can stabilize dynamics with poor initial structures. Runs can be cleanly aborted by "`touch EXIT`". See `Run` command.
- Better handling/reporting of non-converged SCF during MD runs.
- Fixed an issue which slowed down molecular dynamics after many steps.
- Stefan Grimme's xTB method can now be used in the MD module, allowing fast simulations of large systems.

7.1.3 Changes in ORCA 4.1 (Dec 2018)

- Molecular dynamics simulation can now use Cartesian, distance, angle, and dihedral angle constraints. These are managed with the `Constraint` command.
- The MD module now features cells of several geometries (cube, orthorhombic, parallelepiped, sphere, ellipsoid), which can help to keep the system inside of a well-defined volume. The cells have repulsive harmonic walls.
- The cells can be defined as elastic, such that their size adapts to the system. This enables to run simulations under constant pressure.
- Trajectories can now be written in XYZ and PDB file format.
- A restart file is written in each simulation step. With this file, simulations can be restarted to seamlessly continue (useful for batch runs or if the job crashed). Restart is handled via the `Restart` command; see below.
- Introduced regions (*i. e.*, subsets of atoms), which can be individually defined. Regions can be used to thermostat different parts of the system to different temperatures (*e. g.*, cold solute in hot solvent), or to write subset trajectories of selected atoms.
- The energy drift of the simulation is now displayed in every step (in units of Kelvin per atom). Large energy drift can be caused by poor SCF convergence, or by a time step length chosen too large.
- Physical units in the MD input are now connected to their numeric values via underscore, such as `350_pm`. A whitespace between value and unit is no longer acceptable. This slightly breaks backward compatibility – sorry.

- Fixed a bug in the time integration of the equations of motion, which compromised energy conservation.
- Fixed crashes for semiempirics and if ECPs were employed. You can now run MD simulations with methods such as PM3 and with ECPs.

7.1.4 Input Format

The molecular dynamics module is activated by specifying "MD" in the simple input line. The actual MD input which describes the simulation follows in the "%md" section at some later position in the input file. The contents of this section will subsequently be referred to as "MD input".

```
! MD BLYP D3 def2-SVP
%md
  Timestep 0.5_fs # This is a comment
  Initvel 350_K
  Thermostat NHC 350_K Timecon 10.0_fs
  Dump Position Stride 1 Filename "trajectory.xyz"
  Run 200
end
* xyz 0 1
O         -4.54021        0.78439        0.09307
H         -3.64059        0.38224       -0.01432
H         -4.63463        1.39665       -0.67880
*
```

Please note that the MD input is not processed by ORCA's main parser, but by a dedicated parser in the MD module. Therefore, the MD input is not required to obey the general ORCA syntax rules. The syntax will be described in the following.

In contrast to general ORCA input, the MD input is not based on keywords, but on *commands*, which are executed consecutively on a line-by-line basis starting at the top (like, *e. g.*, in a shell script). This means that identical commands with different arguments may be given, coming into effect when the interpreter reaches the corresponding line. This enables to perform multiple simulations (*e. g.*, pre-equilibration and production run) within a single input file:

```
%md
  Timestep 1.0_fs
  Run 200
  Timestep 2.0_fs
  Run 500
end
```

Work is already under way to add variable definitions, loops, and conditional branching to the MD input.[2] This will enable even larger flexibility (*e. g.*, to run a simulation until a certain quantity has converged). The MD input is written in the SANscript language ("Scientific Algorithm Notation Script"), which is under development. A first glimpse can be found at

https://brehm-research.de/sanscript

As in standard ORCA input, **comments** in the MD input are initiated by a "#" sign and span to the end of the current line. Commands can be started both at the beginning of a line and after a command. The only place where a "#" is **not** treated as start of a comment is inside of a string literal (*e. g.*, in file names).

```
%md
  # Comment
  Timestep 0.5_fs # Comment
  Dump Position Filename "trajectory#1.xyz"
end
```

Some more MD input syntax rules:

[2] Technically speaking, ORCA will then be a Turing-complete script interpreter, such that *any* computational problem can be solved with ORCA :-)

- The MD input is generally **not** case-sensitive. The only exception are file names on platforms with case-sensitive file systems (such as GNU Linux).
- Empty lines are allowed.
- Commands and options are separated by space or tabulator characters. Any combination of these characters may be used as separator.
- Both DOS and UNIX line break style is acceptable.

Commands

As already noted above, the central item of the MD input is a command. Each input line contains (at most) one command, and these commands are executed in the given order. A command typically takes one or more arguments, which are given behind the command name, separated by whitespaces, tabulator characters, or commas (optional). The order of the arguments for a command is fixed (see command list in section *Command List*. Commands may have optional arguments, which are always specified at the end of the argument list, after the last non-optional argument. If there exist multiple optional arguments for a command, not all of them need to be specified; however, they need to be specified in the correct order and without gaps:

```
%md
  Command Arg1 Arg2 Arg3                  # fine
  Command Arg1, Arg2, Arg3                # fine
  Command Arg1 Arg2 Arg3 Optarg1          # fine
  Command Arg1 Arg2 Arg3 Optarg1 Optarg2  # fine
  Command Arg1 Arg2 Arg3 Optarg2          # will not work
end
```

Apart from arguments and optional arguments, commands can also have *modifiers*. These can be considered as "sub-commands", which modify a given command, and may possess their own argument lists. Modifiers generally follow after all non-optional and optional arguments, and they may **not** possess optional arguments on their own. If a command has multiple modifiers, the order in which they are given is not important.

In the following input example, "`Mod1`" and "`Mod2`" are modifiers of "`Command`". "`Mod1`" takes one argument, "`Mod2`" does not take arguments:

```
%md
  Command Arg1                            # fine
  Command Arg1 Optarg1                    # fine
  Command Arg1 Mod1 Modarg1 Mod2          # fine
  Command Arg1 Mod2                       # fine
  Command Arg1 Mod2 Mod1 Modarg1          # fine
  Command Arg1 Optarg1 Mod1 Modarg1 Mod2  # fine
end
```

To make this abstract definition a little more illustrative, please consider again one line from the input sample at the beginning of this section:

```
%md
  Dump Position Stride 1 Filename "trajectory.xyz"
end
```

Here, "`Dump`" is the command, which takes one non-optional argument to specify which quantity shall be dumped – in this case, "`Position`". The "`Dump`" command has two modifiers, namely "`Stride`" and "`Filename`". The former takes one integer argument, the latter a string argument. Swapping the two modifiers (together with their respective arguments, of course) would not change the behavior.

Separating Arguments

As shown above, the arguments which are passed to a command do not need to be separated by commas. However, it is allowed (and recommended) to still use commas. First, it can increase the readability of the input file. Secondly, there exist a few ambiguous cases in which commas (or parentheses) should be used to clarify the intended meaning. One of these cases is the arithmetic minus operator. It can either be used as binary operator (subtracting one number from another), or as unary operator (returning the negative of a number). By default, the minus operator will be considered as binary operator (if possible).

Consider the case in which you want to pass two integer arguments "`10`" and "`-10`" to a command. Without commas (or parentheses), the minus is mistreated as binary operator, and only one argument will be passed to the command:

```
Command 10 -10    #  Pitfall: treated as "Command (10 - 10)", i.e., "Command 0"
Command 10, -10   #  Two arguments, as intended
Command 10 (-10)  #  Also works
```

Physical Units

In many cases, it is required to specify quantities which bear a physical unit in an input file (*e. g.*, temperature, time step lengths, …). For many quantities, there are different units in widespread use, which always leads to some confusion (just consider the "kcal vs kJ" case). ORCA handles this problem by defining default units for each quantity and requiring that all quantities are given in their default unit. ORCA's default units are the atomic units, which are heavily used in the quantum chemistry community, but not so much in the molecular dynamics community. As an *ab initio* molecular dynamics module exists in the small overlap region of both communities, some "unit conflicts" might arise. To prevent those from the beginning, it is allowed to specify units of personal choice within ORCA's MD input.

Luckily, this is as simple and convenient as it sounds. The parser of the MD module checks if a unit is given after a numeric constant, and automatically converts the constant to the internal default unit. If no explicit unit is given, the default unit is assumed. Please note that the default units within the MD module are not necessarily atomic units (see table below). Units are connected to the preceding numerical value by an underscore:

```
%md
  Timestep  1.0_fs
  Timestep 41.3_au # identical
  Timestep  1.0    # identical, as default time unit in MD module is fs
end
```

In the following, all units which are currently known to the MD module's parser are listed, sorted by physical quantities. The default unit for each quantity is printed in **`bold letters`**. Additive constant and factor are applied to convert a unit into the default unit. The additive constant is applied before the factor. A "–" means that the constant/factor is not applied. More units will be probably added in the future.

Unit Symbol	Additive Constant	Factor
— Length Units —		
Angstrom	–	–
A	–	–
Bohr	–	0.5291
pm	–	0.01
nm	–	10.0
— Time Units —		
fs	–	–
ps	–	1000
au	–	0.02419
— Temperature Units —		
Kelvin	–	–
K	–	–
Celsius	273.15	–
C	273.15	–
— Angle Units —		
Deg	–	–
Rad	–	$180/\pi$

7.1.5 Discussion of Features

Restarting Simulations

Ab initio molecular dynamics simulation are computationally expensive, and will typically run for a long time even in the case of medium-sized systems. Often, it is desirable to perform such a simulation as a combination of multiple short runs (*e. g.*, if the queuing system of the cluster imposes a maximum job time). The ORCA MD module writes a restart file in each simulation step, which allows for the seamless continuation of simulations. This restart file has the name "`basename.mdrestart`", where `basename` is the project's base name. To load an existing restart file, use the `Restart` command *(see command list below)*.

In the first run of a planned sequence of runs, no restart file exists yet. for this case, the `Restart` command offers the `IfExists` modifier. The restart file is only loaded if it exists. If not, the restart is simply skipped, and no error is thrown. By using this modifier, you can have the `Restart` command already in place in the first run of a sequence (where no restart file exists in the beginning), and do not need to modify the input after the first run has finished.

Concerning the `Dump` command, it is good to know that trajectory files are appended (*not* overwritten) by default. If you ever want to overwrite an existing trajectory file by a `Dump` command, use the `Replace` modifier.

Please note that **only** the positions, velocities, thermostat internal state *(only for NHC)*, Metadynamics hills, and time step counters are restarted when executing a `Restart` command. All other properties (thermostats, regions, trajectory dumps, constraints, cells, etc.) are **not** restarted. They should all remain in the input file, as executed in the first run of a sequence. Just add the `Restart` command after all other relevant commands have been executed, directly before the `Run` command.

To conclude this discussion, a short example is given. If the MD input file

```
%md
  Timestep 0.5_fs
  Initvel 300_K
  Thermostat NHC 300_K Timecon 10.0_fs
  Dump Position Stride 1 Filename "trajectory.xyz"
  Restart IfExists
```

(continues on next page)

(continued from previous page)

```
  Run 100
end
```

is subsequently executed ten times (without any modification), the resulting trajectory file will be identical *(apart from numerical noise)* to that obtained if the following input is executed once:

```
%md
  Timestep 0.5_fs
  Initvel 300_K
  Thermostat NHC 300_K Timecon 10.0_fs
  Dump Position Stride 1 Filename "trajectory.xyz"
  Run 1000
end
```

Regions

In the ORCA MD module, **regions** can be defined. This concept does *not* refer to regions in space, but rather to subsets of atoms in the system. A region is nothing more than a list of atoms. Regions may overlap, *i. e.*, atoms can be part of more than one region at a time. The atoms which are part of a certain region remain the same until the region is manually re-defined, *i. e.*, regions are fixed and do not adapt to any changes in the system. There exist a few pre-defined regions which have a name. User-defined regions, in contrast, only carry an integer identifier. The following regions are pre-defined in any case:

- `all`: Contains all atoms of the system. This is the default if no region is specified in some commands, so by default, these commands will always act on the whole system.
- `active`: This region contains all movable ("non-frozen") atoms. By default, it is identical to the `all` region. Atoms inside of this region are updated by the time integration in a molecular dynamics run, displaced in a minimization, and are considered for computing the kinetic energy.
- `inactive`: This region contains all atoms which are *not* part of the `active` region. These atoms are "frozen"; they are ignored by the time integration / minimization, and also not considered for the computation of the kinetic energy. They simply remain on their initial positions. This is in principle identical to applying Cartesian constraints to the atoms; however, it is much faster. As constraints have to be solved iteratively *(see below)*, Cartesian constraints become quite computationally demanding if applied to thousands of atoms.

From these three pre-defined regions, only the `active` region can be manually modified. Changes in the composition of the `active` region automatically modify the `inactive` region. The `all` region obviously cannot be changed.

In case of a QM/MM simulation, the following four additional regions can be used:

- `qm`: This is the "quantum mechanics" region – it contains all atoms which are treated by the electron structure method.
- `mm`: This is the "molecular mechanics" region – it contains all atoms which are treated by a force-field approach. It exactly contains those atoms which are not part of the `qm` region.
- `active_qm`: Contains exactly those atoms which are part of both the `qm` and the `active` regions.
- `active_mm`: Contains exactly those atoms which are part of both the `mm` and the `active` regions.

These regions can **not** be modified in the MD input. The MD module just reads the region definitions from the QM/MM module, but is not able to make any changes here.

Regions can be useful for many purposes. For example, a "realistic" wall of atoms can be built around the system by defining the `active` region such that it only contains the non-wall atoms. The wall atoms will then be frozen. Apart from that, trajectories of regions can be written to disk, only containing the "interesting" part of a simulation. Furthermore, velocity initialization can be applied to regions, enabling to start a simulation in which different sets of atoms possess different initial temperatures. Thermostats can be attached to regions to keep different sets of atoms at different temperatures during the whole simulation. This allows for sophisticated simulation setups (cold solute in hot solvent, temperature gradient through the system, etc).

Regions are defined or modified by the `Manage_Region` command. Many other commands take regions as optional arguments. Please see the command list below.

Metadynamics

Metadynamics is a powerful tool to analyze free energy profiles of reactions and other processes *(solvation, aggregation, conformer change, dissociation)* based on molecular dynamics simulations. It has been developed by Laio and Parrinello in 2002 [804]. In principle, the frequency of observing a certain process in MD simulations is directly related to the free energy barrier of the process. However, many interesting processes *(such as chemical reactions)* possess such a high free energy barrier that they will never occur on the time scales typical for AIMD simulations (100 ps). To increase the frequency at which such processes happen, so-called rare event sampling methods can be employed. Metadynamics is one among those. It works by building up a bias potential as a sum of Gaussian hills, so that free energy minima are slowly filled up and the system is gradually pushed away from its resting points.

Please note that there is also a method with the same name for exploring conformation space that has been published by Grimme in 2019 [814]. It is in principle based on the original "Parrinello" Metadynamics, but with several modifications and extensions. The ORCA MD module contains the original Parrinello variant of Metadynamics [804], together with several extensions such as well-tempered Metadynamics [808] and extended Lagrangian Metadynamics [805]. The Grimme method for conformer search will probably be implemented in the future.

In Metadynamics, one has to define one or more "collective variables" (Colvars) along which the free energy profile of the system will be sampled. A Colvar is in principle nothing more than a continuous function of all atom positions which returns a real number. A simple example of a Colvar is the distance between two atoms, which could be used to explore the free energy profile of a bond formation or cleavage. In the ORCA MD module, Colvars can be defined via the `Manage_Colvar` command. Available Colvar types are distances *(including projections onto lines or into planes)*, angles, dihedral angles, and coordination numbers [805]. The latter allows, *e. g.*, to accurately compute $\mathrm{p}K_\mathrm{A}$ values of weak acids in solvent [806, 807]. For the distances, angles, and dihedral angles, atom groups instead of single atoms can be specified, so that, *e. g.*, the distance between the centers of mass of two molecules can be defined as a Colvar.

Based on one or two Colvars (ORCA supports one-dimensional and two-dimensional Metadynamics), a Metadynamics simulation can be set up. There are many parameters to choose, which are described in the section of the `Metadynamics` command. After all parameters have been set, the actual simulation is simply started via the `Run` command. It is also possible to restart Metadynamics simulations so that they can be split into multiple successive runs; see the `Restart` command. A full example for a two-dimensional well-tempered extended Lagrangian Metadynamics simulation can be found on *below*.

Note that Metadynamics simulations typically require very much computational time *(at least several 10 000 MD steps for a roughly converged result, depending on the Colvar choice)*. So this is by no means a method to "shortly try out". However, there are no cheaper methods for predicting free energy profiles *(apart from very simple approximations such as the harmonic oscillator)*, and the predictive power of computing free energy profiles comes at a price.

7.1.6 Command List

In Table 7.1, an alphabetical list of all commands currently known to the MD module is given. The description of each command starts with a small box which contains the command's name and a table of arguments and modifiers. The last-but-one column in the table specifies the type of each argument. Possible types are "Integer", "Real", "String", and "Keyword". In the latter case, the last column contains a list of allowed keyword values in { `braces` }. If the type is "Real" and is a physical quantity with unit, the quantity is given in the last column in [square brackets]. Each such box is followed by a textual description of the corresponding command.

Table 7.1: Overview of commands in the `%md` block

Command	Description
`Cell`	Defines and modifies cells
`Constraint`	Manages constraints
`Dump`	Controls trajectory output
`Initvel`	Randomly initializes atom velocities
`Manage_Colvar`	Manages collective variables ("Colvars")
`Manage_Region`	Manages regions
`Metadynamics`	Sets parameters for Metadynamics runs
`Minimize`	Performs a Cartesian energy minimization
`PrintLevel`	Controls the output verbosity
`Randomize`	Sets the random seed
`Restart`	Restarts a simulation to seamlessly continue
`Restraint`	Manages restraints on Colvars
`Run`	Performs a molecular dynamics run
`SCFLog`	Controls the ORCA log file output
`Screendump`	Prints current MD state to screen
`Thermostat`	Manages thermostats
`Timestep`	Sets the integrator time step Δt

`Cell`

Manadory Arguments:		-		
Optional Arguments:		-		
Modifiers:	`Cube`	...	...	*see text*
	`Rect`	...	...	*see text*
	`Rhomb`	...	...	*see text*
	`Sphere`	...	...	*see text*
	`Ellipsoid`	...	...	*see text*
	`None`	—	—	—
	`Spring`	k	Real	*see text*
	`Elastic`	t_{avg}	Real	[time]
		c_{response}	Real	*see text*
	`Anisotropic`	—	—	—
	`Pressure`	...	...	*see text*
	`Fixed`	—	—	—

Defines a harmonic repulsive wall around the system *(the wall is "soft" with a spring constant and atoms can slightly penetrate; "hard" repulsive walls are not supported)*. This helps to keep the molecules inside of a well-defined volume, or to keep a constant pressure in the system. In the latter case, the cell can be defined as elastic, such that it exerts a well-defined pressure *(see below)*. Please note that ORCA does not feature periodic boundary conditions, and therefore, all cells are non-periodic (just repulsive walls). There are several cell geometries available *(only one type of cell can be active at a time)*:

- **`Cube`**: Defines a cubic cell. If two real values p_1 and p_2 are specified as coordinates, the cell ranges from (p_1, p_1, p_1) to (p_2, p_2, p_2). If only one real value p is supplied, the cell ranges from $\left(-\frac{p}{2}, -\frac{p}{2}, -\frac{p}{2}\right)$ to $\left(\frac{p}{2}, \frac{p}{2}, \frac{p}{2}\right)$, *i. e.* it is centered at the origin with edge length p.
- **`Rect`**: Defines an orthorhombic cell. Six real values x_1, y_1, z_1, x_2, y_2, and z_2 have to be specified as coordinates *(in this order)*. The cell will range from (x_1, y_1, z_1) to (x_2, y_2, z_2).
- **`Rhomb`**: Defines a parallelepiped-shaped cell (also termed as rhomboid sometimes). You have to specify twelve real values in total. The first three define one corner point p of the cell, and the remaining nine define three cell vectors v_1, v_2, and v_3, each given as Cartesian vector components. The cell is then defined as the set of points $\{p + c_1 v_1 + c_2 v_2 + c_3 v_3 \mid 0 \leq c_1, c_2, c_3 \leq 1\}$ The vectors v_1, v_2, and v_3 do not need to be orthogonal to each other, but they may not all lie within one plane (cell volume would be zero).

- **`Sphere`**: Defines a spherical cell. You need to specify four real values c_x, c_y, c_z, and r. The cell will then be defined as a sphere around the central point $\left(c_x, c_y, c_z\right)$ with radius r.
- **`Ellipsoid`**: Defines an ellipsoid-shaped cell. As first three arguments, you have to specify three real values c_x, c_y, c_z, which define the center of the ellipsoid to be $\left(c_x, c_y, c_z\right)$. As fourth argument, a keyword has to follow, which may either be "`XYZ`" or "`Vectors`". In the "`XYZ`" case, three more real values r_x, r_y, and r_z have to be specified, which define the partial radii of the ellipsoid along the X, Y, and Z coordinate axes. If instead "`Vectors`" was given, nine more real values v_x^1, v_y^1, v_z^1, v_x^2, v_y^2, v_z^2, v_x^3, v_y^3, v_z^3 have to follow after the keyword. These values define three vectors $v^1 := \left(v_x^1, v_y^1, v_z^1\right)$, $v^2 := \left(v_x^2, v_y^2, v_z^2\right)$, and $v^3 := \left(v_x^3, v_y^3, v_z^3\right)$, which are the principal axes of the ellipsoid. These vectors have to be strictly orthogonal to each other. The length of each vector defines the partial radius of the ellipsoid along the corresponding principal axis.

All cell types define a harmonic potential $E_{\text{cell}}\left(r\right) := k \cdot r^2$ which acts on all atoms in the system **outside of the cell**, where r is the closest distance from the atom's center to the defined cell surface. Atoms whose center is inside of the cell or directly on the cell surface do not experience any repulsive force. Following from the definition, the force which acts on an atom outside of the cell is always parallel to the normal vector of the cell surface at the point which is closest to the atom center. This is trivial in case of cubic, rectangular, rhombic, and spherical cells, but not so trivial for ellipsoid-shaped cells.

The spring constant k in the above equation (*i. e.*, the "steepness" of the wall) can be specified by the "`Spring`" modifier, which expects one real value as argument. The spring constant has to be specified in the unit kJ mol^{-1} Å^{-2}, other units cannot be specified here. The default value is 10 kJ mol^{-1} Å^{-2}. Larger spring constants reduce the penetration depth of atoms into the wall, but may require shorter integration time steps to ensure energy conservation. If jumps in the total energy occur, try to use a smaller spring constant (*e. g.*, the default value).

The command "`Cell None`" disables any previously defined cell.

If you want to perform simulations under constant pressure, you can define an elastic cell. Then, ORCA accumulates the force which the cell exerts on the atoms in each time step, and divides this total force by the cell surface area to obtain a pressure. As this momentarily pressure heavily fluctuated, a running average is used to smooth this quantity. If the averaged pressure is larger than the external pressure which was specified, the cell will slightly grow; if it was smaller, the cell will slightly shrink. In the beginning of a simulation, the cell size will not vary until at least the running average history depth of steps have been performed.

An elastic cell is enabled by using the "`Elastic`" modifier after the cell geometry definition. Subsequently, two real values t_{avg} and c_{response} are required. While t_{avg} defines the length of the running average to smooth the pressure *(in units of physical time, **not** time steps)*, the c_{response} constant controls how fast the cell size will change at most. More specific, c_{response} is the fraction of the cell volume growth per time step if the ratio of averaged and external pressure would be infinite, and at the same time the fraction of the cell volume reduction per step if the aforementioned ratio is zero. Put into mathematical form, the cell volume change per time step is

$$V_{\text{new}} := \begin{cases} V_{\text{old}} \cdot \frac{c_{\text{response}} \cdot \frac{\langle p \rangle}{p_{\text{ext}}} + 1}{c_{\text{response}} + 1} & \text{if } \frac{\langle p \rangle}{p_{\text{ext}}} \leq 1, \\ V_{\text{old}} \cdot \frac{\left(c_{\text{response}} + 1\right) \cdot \frac{\langle p \rangle}{p_{\text{ext}}}}{\frac{\langle p \rangle}{p_{\text{ext}}} + c_{\text{response}}} & \text{if } \frac{\langle p \rangle}{p_{\text{ext}}} > 1, \end{cases} \tag{7.1}$$

where $\langle p \rangle$ represents the averaged pressure the system exerts on the walls, and p_{ext} is the specified external pressure. Good starting points are $t_{\text{avg}} = 100\,\text{fs}$ and $c_{\text{response}} = 0.001$. Please note that larger values of c_{response} or smaller values of t_{avg} may lead to uncontrolled fluctuations of the cell size. An already defined fixed cell can be switched to elastic by the command "`Cell Elastic ...`" (the dots represent the two real arguments).

By default, the size change of an elastic cell due to pressure is performed *isotropically*, *i. e.*, the cell is scaled as a whole, and exactly retains its aspect ratio. By specifying the "`Anisotropic`" modifier after switching on an elastic cell, the cell pressure is broken down into individual components, and the size of the cell is allowed to change independently in the individual directions. This, of course, only makes sense for the cell geometries `Rect`, `Rhomb`, and `Ellipsoid`. An already defined isotropic cell can be switched to anisotropic by simply executing "`Cell Anisotropic`".

In case of an elastic cell, the external pressure is defined by the modifier "`Pressure`", which expects either one or three real values as arguments. If one argument is given, this is the isotropic external pressure. If three arguments are supplied, these are the components of the pressure in X, Y, and Z direction *(in case of orthorhombic cells)* or along the direction of the three specified vectors *(in case of parallelepiped-shaped and ellipsoid-shaped cells)*. This allows for anisotropic external pressure (probably only useful for solid state computations). Both the pressure and the pressure

components have to specified in units of bar ($= 10^5$ N m^{-2}), other units cannot be used. If this modifier is not used, the default pressure will be set to 1.0 bar (isotropic) if an elastic cell is used. The external pressure of an already defined cell can be changed by the command "`Cell Pressure ...`" (the dots represent the real argument(s)).

As all cells are non-elastic by default, there is no keyword to explicitly request this at the time of cell definition. However, possible applications might require to use an elastic cell during equilibration period, and then "freeze" this cell at the final geometry for the production run. This can be achieved by using the "`Cell Fixed`" command (without any additional arguments).

If the cell is elastic, there is a volume work term which contributes to the total energy of the system. ORCA computes this term in every step and adds it to the potential energy. Without this contribution, the conserved quantity would drift excessively in elastic cell runs.

To completely switch off a previously defined cell, simply use "`Cell None`".

Please note that cells are **not** automatically restarted by using the `Restart` command.

Examples:

Cubic cell with edge length 10 Å centered around origin:
`Cell Cube 10`

Spherical cell with radius 5 Å centered around origin and 20 kJ mol^{-1} Å^{-2} wall steepness:

```
Cell Sphere 0, 0, 0, 5 Spring 20
```

Elastic orthorhombic cell from $(-2, -2, 0)$ to $(12, 12, 10)$, $t_{\text{avg}} = 100$ fs, $c_{\text{response}} = 0.001$:

```
Cell Rect -2, -2, 0, 12, 12, 10 Elastic 100, 0.001
```

Ellipsoid-shaped cell centered on origin with partial radii 5, 10, 15 Å along the X, Y, Z axes:

```
Cell Ellipsoid 0, 0, 0 XYZ 5, 10, 15
```

The commas are optional, but make sure to use them with negative numbers. By default, the minus operator will act as binary operator if possible *(see* **discussion above***)*.

`Constraint`

Mandatory Arguments:		*operation*	Keyword	{ `Add`, `Remove`, `List` }
		type	Keyword	{ `Cartesian`, `Distance`, `Angle`, `Dihedral`, `Center`, `Rigid` }
		atom(s)	Integer	—
Optional Arguments:		—		
Modifiers:	`Target`	*value(s)*	Real	—
	`Ramp`	*value(s)*	Real	—
	`Weights`	...	...	*see text*
	`All`	—		
	`Noprint`	—		

Manages constraints in the molecular dynamics simulation. Unlike `Restraints`, constraints are geometric relations which are strictly enforced at every time (*i. e.*, they do not fluctuate around their target value). All atoms involved in constraints have to be included in the active region. In principle, constraints also work in Cartesian geometry optimizations with the `Minimize` command, but the performance together with L-BFGS may be poor (except for Cartesian constraints, which work flawlessly in L-BFGS). In these cases, try to use the simulated annealing method instead.

The simplest possibility is to constrain the Cartesian position of an atom to some value. A zero-based atom index is required. The command `Constraint Add Cartesian 3` would fix the fourth atom in the simulation at its current position in space. If the desired position shall be explicitly given, it can be specified via the `Target` modifier, *e. g.*, `Constraint Add Cartesian 3 Target 5.0 1.0 1.0`. To determine which dimensions to fix, one of the `XYZ`, `XY`, `XZ`, `YZ`, `X`, `Y`, or `Z` modifiers can be added. For example, `Constraint Add Cartesian`

`3 X Target 1.0` would constrain the X coordinate of atom 3 to the absolute value 1.0, but would not influence movement along the Y and Z coordinate at all.

By using the `Distance` keyword, distances between atoms can be fixed. The command `Constraint Add Distance 3 5` would fix the distance between atom 3 and 5 to its current value. You need to specify exactly two atom indices; multiple distance constraints are entered via multiple `Constraint` commands. Also here, a desired distance value can be given via the `Target` modifier, such as `Constraint Add Distance 3 5 Target 350_pm`.

Similarly, angles and dihedral angles between atoms can be fixed with the `Angle` and `Dihedral` keywords. Angles are defined by three atom indices, and dihedral angles by four atom indices. Also here, target values may be specified. Any combination of Cartesian, distance, angle, and dihedral constraints may be used simultaneously, and may even be applied to the same group of atoms. A molecule can be made completely rigid by constraining all its bonds, angles, and torsions. Please make sure that your constraints are not over-determined, and do not contradict each other. Otherwise, they can't be enforced and the simulation will print warnings or crash.

A different and powerful class of constraints can be defined with the `Center` argument. Directly after the keyword, a list of integer atom numbers is expected. This list can be a combination of numbers and ranges, *e. g.*, "`1, 3, 5..11, 14`". The weighted average position of this subset of atoms is then constrained to a fixed position in Cartesian space. By default, the weights are taken as the atom masses, such that the center of mass of the selected atoms is kept fixed. This allows, *e. g.*, to run a MD simulation of two molecules with fixed center of mass, such that their center of mass distance remains constant. Custom weights for the definition of the center can be entered by using the `Weights` modifier after the atom list. It expects exactly the same number of real arguments as the length of the specified atom list. The geometric center of a group of atoms can be held fixed by setting all weights to 1.0, for example "`Constraint Add Center 2, 5..7 Weights 1.0 1.0 1.0 1.0`". If desired, a `Target` for the center position can be given, which expects three real numbers for the X, Y, and Z coordinate after the keyword. If no target is specified, the current center position is held fixed.

With the `Rigid` type of constraints, complete groups of atoms can be kept rigid, *i. e.*, keep all their distances and angles relative to each other, but move as a whole. After the `Rigid` keyword, a list of atom numbers is expected. More than one group of atoms can be kept rigid at the same time – just call the `Constraint Add Rigid` command multiple times with different atom lists. Internally, the rigid constraint is realized by defining the correct number of distance constraints. Such a large number of distance constraints is hard to converge; therefore, warning messages that RATTLE did not converge will **not** be shown if a rigid constraint is active. Almost planar (or even linear) groups of atoms are hard to keep rigid by using only distance constraints. It might help do add a dummy atom outside of the plane and include this into the constraint.

ORCA supports constraints with linearly changing target value during the simulation. To define such a constraint, write "`Ramp`" directly after the "`Target`" modifier. After "`Ramp`", twice the number of real numbers that would have been required for "`Target`" follows (two instead of one for distances, angles, and dihedrals; six instead of three for "`Cartesian XYZ`", and so on). The first half of these arguments are the starting values, the second half are the final target values. For example, "`Constraint Add Distance 3 5 Target Ramp 300_pm 400_pm`" will define a distance constraint with a target value rising from 300 pm to 400 pm. The ramp will be performed once during the *`Run`* command which follows next after the constraint definition. Therefore, the number of steps specified in this *`Run`* command also specifies the rate at which the constraint target is modified. After the ramp has been completed once, the final (constant) target value(s) will be used for all subsequent *`Run`* commands.

If an already defined constraint is defined again, it is overwritten, *i. e.*, the old version of the constraint is automatically deleted first.

Constraints are removed with the `Remove` keyword. You can either remove single constraints, *e. g.*, `Constraint Remove Distance 3 5`, or groups of similar constraints. To remove all angle constraints, use `Constraint Remove Angle All`. To remove all restraints, enter `Constraint Remove All`.

The `List` argument prints all currently active constraints to the screen and log file. No additional arguments can be specified.

By default, the external force acting on each constraint is computed in every MD step and written to a file named "`basename-constraints.csv`" *(one column per constraint)*. This can be useful – the average force acting on a constraint can be, *e. g.*, used for thermodynamic integration [813]. If a large number of constraints is defined, this might waste computer time if it is not required. In these cases, the constraints can be defined with the `Noprint` modifier. For such constraints, the acting forces are not computed and not written to the file. Note that constraints

which have been pre-defined *(e. g., by the force field for rigid molecules such as TIP3P water)* automatically have this modifier.

Please note that each constraint decreases the number of the system's degrees of freedom (DoF) by one. This effect is included, *e. g.*, in the temperature computation, where the DoF count enters. From this consideration, it can also be understood that a constraint behaves significantly different from a restraint with very large spring constant: In the former case, the DoF is removed from the system; in the latter case, the DoF is still there, but can only move in a tiny interval.

It is computationally inefficient to define a large number of Cartesian constraints if a subset of atoms simply shall be fixed. A more efficient approach is to define an active region which only contains the atoms which shall be movable (see `Manage_Region` command). All atoms outside of the active region will not be subject to time integration and therefore keep their positions. However, please note that these atoms may not be involved in any other (distance, angle, dihedral) constraint.

`Dump`

Mandatory Arguments:		*quantity*	Keyword	{ `Position`, `Velocity`, `Force`, `GBW`, `EnGrad` }
Optional Arguments:		—		
Modifiers:	`Format`	*fmt*	Keyword	{ `XYZ`, `PDB`, `DCD` }
	`Stride`	*n*	Integer	—
	`Filename`	*fname*	String	—
	`Region`	*region*	…	…
	`Replace`	—		
	`None`	—		

Specifies how to write the output trajectory of the simulation. The *quantity* argument can be one of the keywords `Position`, `Velocity`, `Force`, `GBW`, and `EnGrad`. While the velocities are written in Angstrom/fs, the unit of the forces is Hartree/Angstrom. The following paragraphs only apply to the first three quantities. Dumping GBW and EnGrad files works differently, and is described at the very end of this section.

The `Stride` modifier specifies to write only every n-th time step to the output file (default is $n = 1$, *i. e.*, every step). A stride value of zero only writes one frame to the trajectory at the time when the `Dump` command is called – no further frames will be written during the run. This can be helpful, *e. g.*, to write an initial PDB snapshot for DCD trajectories, or to keep a single GBW file at some point.

The `Format` modifier sets the format of the output file. Currently, only the `XYZ`, `PDB`, and `DCD` formats are implemented. Please note that the `DCD` format is not well-defined, and different programs use different formats with this extension. Furthermore, `DCD` files do not store atom type information and are only valid together with a PDB snapshot of the system (a single PDB snapshot can be written via "`Dump Position Format PDB Stride 0`"). If not specified, ORCA tries to deduce the format from the file extension of the specified file name. If also no file name is given, trajectories will be written in XYZ format by default.

The `Filename` modifier sets the output file name. If not specified, the default file name will have the form "`proj-qty-rgn.ext`", where `proj` is the base name of the ORCA project, `qty` is one of `postrj`, `veltrj`, or `frctrj`, `rgn` specifies the name or number of the region for which the dump is active, and `ext` is the file extension selected by the `Format` modifier.

If the trajectory file already exists at the beginning of a `Run` command, new frames will be appended to its end by default. If you want to overwrite the existing file instead, use the `Replace` modifier. The old existing file is erased only once after a dump with this modifier has been specified. If multiple `Run` commands follow after the dump definition, the trajectory will **not** be replaced before each of these runs, only before the first one among them. To overwrite the file another time, simply re-define the dump with the `Replace` modifier. If the file does not yet exist at the beginning of a run, this modifier has no effect. Appending frames to DCD trajectories is not possible *(because they store the total frame count in the header)*. Therefore, `Replace` is automatically switched on if the format is DCD.

With the `Region` modifier, the trajectory output can be restricted to a specific region (*i. e.*, subset of atoms). This modifier expects one argument, which is either the name of a pre-defined region or the number of a user-defined region *(see above)*. If not specified, the trajectory of the whole system will be written. Multiple dump commands for multiple regions can be active at the same time, but each pair of region and quantity *(position/velocity/force)* can have only one attached dump command at a time (re-defining will overwrite the dump settings).

Use the `None` modifier to disable writing this quantity to an output file. The command "`Dump Position None`" will disable writing of all position trajectories for all regions. To disable only the dump for a specific region, use "`Dump Position Region r None`", where `r` is the name or number of the region.

The default is to write a position trajectory with `Stride 1` and `Format XYZ` to a file named "`proj-postrj-all.xyz`", where "proj" is the base name of the ORCA project. If you want to create no output trajectory at all, use "`Dump Position None`" as described above.

The `Dump GBW` command keeps a copy of the GBW file every n steps, which can be used for computing properties along the MD trajectory, *e. g.*, plotting orbitals. This does not yield a trajectory, as all the GBW files are stored individually. The value of n is controlled by the `Stride` modifier. The file names are formed by appending the step number *(six digits with leading zeros)* followed by ".gbw" to the `Filename` argument. Therefore, this argument should not contain the ".gbw" extension by itself. If the `Filename` modifier is not specified, the default will be "proj-step", where "proj" is the base name of the ORCA project. This will lead to files such as "`proj-step000001.gbw`", etc. The `Format` and `Region` modifiers can not be used for `Dump GBW`.

In a very similar way, `Dump EnGrad` stores an ORCA `.engrad` file *(energy and gradient)* every n steps. All the `.engrad` files are stored individually *(not as a continuous trajectory)*. The value of n is controlled by the `Stride` modifier. By default file names such as "`proj-step000001.engrad`" will be used.

Initvel

Mandatory Arguments:		*temp*	Real	[temperature]
Optional Arguments:		—		
Modifiers:	`Region`	*region*	…	…
	`No_Overwrite`	—		

Initializes the velocities of the atoms by random numbers based on a Maxwell–Boltzmann distribution, such that the initial temperature matches $temp$ (see also section *1.5.2*). Please note that this overwrites all velocities, so do not call this command when your system is already equilibrated (*e. g.*, to change temperature – use a thermostat instead).

The total linear momentum of the initial configuration is automatically removed, such that the system will not start to drift away when the simulation begins. This only concerns the initial configuration. Total linear momentum might build up during the simulation due to numeric effects.

With the `Region` modifier, the initialization of velocities can be performed for a specific region (*i. e.*, subset of atoms). This modifier expects one argument, which is either the name of a pre-defined region or the number of a user-defined region *(see above)*. If not specified, the command acts on the whole system.

The `No_Overwrite` modifier only initializes the velocities if no atom velocities have been defined/read before. This is useful in combination with the `Restart` command: After reading an existing restart file, the velocities are already known, and the initialization will be skipped if this modifier is used. The following combination of commands in a MD input would initialize the velocities only upon first execution, and restart the positions and velocities on all following executions of the same input:

```
Restart IfExists
Initvel 350_K No_Overwrite
```

If neither the `Initvel` command nor a *`Restart`* command is not invoked before a *`Run`* call, the atom velocities will be initialized to zero before starting the run.

Manage_Colvar

Mandatory Arguments:		*operation*	Keyword	{ Define }
		id	Integer	—
		type	Keyword	{ Distance, Angle, Dihedral, CoordNumber }
Optional Arguments:		—		
Modifiers:	Atom	*atom*	Integer	—
	Group	*atomlist*	Integers	—
	Weights	*weights*	Reals	—
	Cutoff	*cutoff*	Real	[length]
	Noprint	—		

Defines collective variables ("Colvars") which are used for Metadynamics or to impose Restraints on the system. In a general sense, a Colvar is simply a continuous function of all the atom positions which returns a real number. As Colvars don't have any effect on the simulation by themselves, they can currently only be defined or re-defined; there is no requirement for deleting them. The second argument of the Manage_Colvar command is the number of the Colvar. This number is used to address the Colvar later. Allowed numbers are within the range of $1 \ldots 10000$. If a Colvar number which had previously been defined is defined again, it is simply overwritten *(and all restraints based on the old Colvar are deleted!)*. The third mandatory argument is the type of the Colvar, which can be Distance, Angle, Dihedral, and CoordNumber. More Colvar types will probably be added in the future *(feel free to make suggestions in the forum!)*.

Distance Colvars are defined between two points in space. Each point can either be a single atom (expressed by "Atom") or the weighted average (center) of the positions of a group of atoms (expressed by "Group"). For example, the command "Manage_Colvar Define 1 Distance Atom 0 Atom 7" defines Colvar 1 to be the distance between atoms 0 and atom 7 *(as always, the atom count starts at zero)*. On the other hand, the command "Manage_Colvar Define 2 Distance Group 0 1 2 Group 3 4 5" sets Colvar 2 to be the distance between the centers of atoms 0, 1, 2 and atoms 3, 4, 5. If many atoms shall be selected, the range syntax "Group 0..2" can be used, including multiple such ranges if required, such as in "Group 0..2, 5, 7..11" *(see also discussion of the* Manage_Region *command)*. By default, the center of mass is used for groups. However, weights can be manually specified if required by using the "Weights" modifier directly after the atom list for the center is finished. "Weights" expects as many real numbers as the group possesses atoms, for example "Manage_Colvar Define 2 Distance Atom 0 Group 1 2 3 Weights 1.0 1.0 1.0". The "Atom" and "Group" syntax can be mixed, *e. g.*, to define the distance between a single atom and a center of mass. When defining distance Colvars, one of the modifiers X, Y, Z, XY, XZ, YZ, and XYZ may be specified directly after "Distance". The first three among them denote that the positions shall be projected onto the corresponding Cartesian vector before computing the distance. The following three modifiers require that the two positions are projected into the corresponding Cartesian plane prior to computing the distance. The last one is the default *(just measure the standard distance in 3D space)* and does not need to be specified explicitly.

In a very similar manner, angle Colvars can be defined. Instead of two points in space, an angle Colvar is defined via three points in space, each of which can either be an "Atom" or a "Group" *(see above)*. For example, the command "Manage_Colvar Define 3 Angle Group 0 1 2 Atom 3 Atom 4" defines Colvar 3 to be the angle spanned by the mass center of atoms 0, 1, 2, atom 3, and atom 4, respectively.

Dihedral Colvars are defined through four points in space, each of which can either be an "Atom" or a "Group" *(see above)*. For example, the command "Manage_Colvar Define 4 Dihedral Atom 0 Group 1..5 Atom 6 Atom 7" defines Colvar 4 to be the dihedral angle spanned by atom 0, the mass center of atoms 1, 2, 3, 4, 5, atom 6, and atom 7, respectively.

The Colvar type "CoordNumber" has been suggested in literature [805] to measure the coordination number of some atom species around some other atom. An example where this type of Colvar has been successfully applied is the calculation of $\mathrm{p}K_\mathrm{A}$ values of weak acids in solvent via Metadynamics [806, 807]. The Colvar is defined by the following equation

$$C := \frac{1}{N_A} \sum_i^{N_A} \sum_j^{N_B} \frac{1 - \left(\frac{r_{ij}}{r_\mathrm{cut}}\right)^6}{1 - \left(\frac{r_{ij}}{r_\mathrm{cut}}\right)^{12}},$$

where N_A is the set of atoms which is coordinated *(typically only one atom)*, N_B is the set of coordinating atoms, r_{ij} is the distance between atoms i and j, and r_{cut} is a constant cutoff distance which specifies a threshold for coordination. After "`CoordNumber`", two atoms or groups of atoms must follow, which correspond to N_A and N_B, respectively. The distance cutoff is specified after the "`Cutoff`" modifier which should follow the two group definitions. For example, the command "`Manage_Colvar Define 5 CoordNumber Atom 0 Group 1..10 Cutoff 200_pm`" defines Colvar 5 as the coordination number of the group of atoms 1 to 10 around atom 0 with a distance cutoff of $r_{\text{cut}} = 200$ pm.

For every defined Colvar, the temporal development of the position and the external force acting on the Colvar is written to a text file named "`basename-colvars.csv`" in every MD step by default. If a large number of Colvars are defined, this might be a waste of time and disk space. In these cases, the "`Noprint`" modifier can be specified when defining the Colvar. Colvars defined with this modifier will not appear in the text file, and the force acting on the Colvar will not be computed *(if not required otherwise, e. g., for restraints)*.

Manage_Region

Mandatory Arguments:		*identifier*	Keyword/Integer	...
		operation	Keyword	{ `Define`, `AddAtoms`, `RemoveAtoms` }
		atomlist	Integer(s)	—
Optional Arguments:		—		
Modifiers:	`Element`	*elem*	String	—

Defines or modifies regions. Regions are just subsets of atoms from the system – see [**Section** *1.3*](#moldyn:sec_regions) above.

As described above, there exist several pre-defined regions which are identified by names. The only such pre-defined region which can be re-defined by the user is the `active` region. All atoms in this region are subject to time integration in molecular dynamics and displacement in minimization runs. All other atoms are simply ignored and remain on their initial positions. Please note that the `active` region may never be empty.

To re-define the `active` region, use the command "`Manage_Region active Define 1 5 7 ...`". The integer arguments after `active` are the numbers of the atoms to be contained in the region, in the order given in the ORCA input file. Atom numbers are generally zero-based in ORCA, *i. e.*, counting starts with 0.

Apart from that, user-defined regions are supported. These are identified with an integer number instead of a name. The integer numbers do not need to be sequential, *i. e.*, it is fine to define region 2 without defining region 1. To give an example, the command "`Manage_Region 1 Define 17 18 19`" defines region 1, and adds atoms 17, 18, and 19 to this newly defined region. Using `Define` without an atom list, such as in "`Manage_Region 1 Define`", deletes the user-defined region, as it will be empty then. Atoms can be added to or removed from previously defined regions (including the `active` region) with the `AddAtoms` and `RemoveAtoms` operations. The atom numbers specified after the operation name are added to or removed from the region. For example, "`Manage_Region active RemoveAtoms 15 16 17`" will remove atoms 15 to 17 from the `active` region (and add them to the `inactive` region instead).

If you want to specify a range of atoms, you can use the syntax "`a..b`" to include all atom numbers from `a` to `b`. If you want only, *e. g.*, every third atom in a range, you can use "`a..b..i`" to add the range from `a` to `b` with increment `i`. As an example, "`2..10..3`" will expand to the list `2, 5, 8`. You can mix atom numbers and ranges, as shown in the following two examples *(as always, the commas are optional)*:

```
Manage_Region active Define 1, 4, 5..11, 14, 17..30..2
Manage_Region active RemoveAtoms 4, 15..17
```

Instead of an atom list, the `Element` modifier can be used, followed by a string which represents an element label. This will have the same effect as specifying an atom list with all atoms of this element type instead. Don't forget the double quotes around the element label string. For example, `Manage_Region active RemoveAtoms Element "H"` removes all hydrogen atoms from the active region.

Metadynamics

Mandatory Arguments:		—		
Optional Arguments:		—		
Modifiers:	Off	—		
	Reset	—		
	Colvar	*colvar*	Integer	—
	Scale	*scale*	Real	...
	Wall	*side*	Keyword	{ Lower, Upper }
		target	Real	phys. unit
		k	Real	kJ mol^{-1} phys. $unit^{-2}$
	HillSpawn	*frequency*	Integer	—
		height	Real	kJ mol^{-1}
		sigma	Real	phys. unit
	Range	*from*	Real	phys. unit
		to	Real	phys. unit
		resolution	Integer	—
	Store	*store*	Integer	—
	Temperature	*temp*	Real	[temperature]
	WellTempered	*biastemp*	Real	[temperature]
	Lagrange	*mass*	Real	a.m.u.
		k	Real	kJ mol^{-1} $scaleunit^{-2}$
		target_temp	Real	[temperature]
		tau	Real	[time]

Sets the parameters for a *Metadynamics* simulation [804]. After all parameters have been set, the actual simulation can be started by a *Run* command. The parameters can either all be set in a single call to the Metadynamics command, or distributed over multiple such calls to avoid very long lines. In both cases, there are some rules for the order of parameter settings. All Colvars for the Metadynamics need to be specified before setting any other parameters. Modifiers which are related to Colvars (such as Scale, Wall, or Range) only apply to the Colvar that was specified last before them in the Metadynamics command.

The Colvar modifier specifies a Colvar to be used in the Metadynamics simulation. It expects one integer argument, which is the number of the Colvar, as defined before via the *Manage_Colvar* command. The ORCA MD module supports one- and two-dimensional Metadynamics, so either one or two Colvar modifiers can be given. The number of Colvar modifiers specified defines the dimensionality of the Metadynamics simulation. Please note that modifiers which are related to Colvars (such as Scale, Wall, or Range) only apply to the Colvar that was specified last before them, so after specifying the first Colvar, these should be set before specifying the second Colvar.

Colvars can have different physical units, such as Angstrom for distances and degree for angles. In a multi-dimensional Metadynamics run, the different numerical magnitude of the corresponding numbers can be an issue: Angles span over a range of 180 degree, while distances will often be within an interval of only 10 Angstrom. To bring all Colvars to a similar scale, the Metadynamics module internally divides every Colvar by a user-supplied constant. These internal values are dimensionless, they will be referred to as "scale units". For a previously specified Colvar, the scale constant can be set via the Scale modifier. It expects one real argument which has to be specified in physical units of the Colvar (length units for distance Colvars, angle units for angle and dihedral Colvars). Coordination number Colvars are dimensionless anyway. If not specified, reasonable default values for the scale are used, which are 1.0 Angstrom for distance Colvars, 20.0 degree for angle and dihedral Colvars, and 0.2 for coordination number Colvars. Note that the Scale modifier only applies to the Colvar given last before it in the Metadynamics command.

As an example, consider the following commands to set up a two-dimensional Metadynamics simulation:

```
Manage_Colvar Define 1 Distance Atom 0 Atom 1
Manage_Colvar Define 2 Angle Atom 0 Atom 1 Atom 2
Metadynamics Colvar 1 Scale 1.0_A Colvar 2 Scale 10.0_Deg
```

To keep Colvars within the region of interest during Metadynamics simulations, harmonic walls can be imposed on Colvars. This is achieved via the Wall modifier. As a first argument, it expects the direction of the wall, which can be Lower or Upper. The second argument is the position of this wall – given in physical units of the Colvar (*e.*

g., Angstrom for distance Colvars), **not** in scale units. As an optional third real argument, the spring constant of the harmonic wall can be specified in kJ mol^{-1} unit^{-2}, where unit is the default physical unit of the Colvar (Angstrom for distances, degree for angles). If omitted, a spring constant of 50 kJ mol^{-1} Angstrom^{-2} for distances, 0.5 kJ mol^{-1} degree^{-2} for angles, and 250.0 kJ mol^{-1} for coordination numbers is used *(a reasonable choice)*. Both lower and upper wall can be defined after one `Wall` modifier, such as in "`Metadynamics Colvar 1 Wall Lower 3.0_A 50.0 Upper 10.0_A 50.0`". Note that one-sided harmonic walls can also be imposed on Colvars via the *`Restraint`* command. For standard Metadynamics, this is redundant. However, for extended Lagrangian Metadynamics *(see below)*, it makes a difference: Restraints act on the Colvar and therefore on the real atomistic system, whereas the walls defined in the `Metadynamics` command act directly on the virtual particle. Again, note that the `Wall` modifier only applies to the Colvar given last before it in the `Metadynamics` command.

The last modifier which applies to Colvars is the `Range` modifier. It has no influence on the Metadynamics run itself, and only controls the output of the free energy profiles. The `Range` modifier expects three arguments. The first two have to be real numbers and define the lower and upper interval borders for which the free energy profile with respect to this Colvar shall be output. The third argument is of integer type and controls the number of grid points to produce for this interval. In two-dimensional Metadynamics, both Colvars can have associated `Range` modifiers, which then control the interval and resolution of the 2D grid for the free energy profile. In this case, the grid should not be much finer than 100×100; otherwise, the evaluation of the grid points will become quite slow. If no `Range` modifier is given, default values are used (range of $0 \ldots 20$ Angstrom for distance Colvars, $0 \ldots 180$ degree for angle Colvars, $-180 \ldots 180$ degree for dihedral Colvars, and $0 \ldots 3$ for coordination number Colvars). As above, note that the `Range` modifier only applies to the Colvar given last before it in the `Metadynamics` command.

Addition of new Gaussian hills to the bias potential is controlled via the `HillSpawn` modifier. It expects three arguments. The first argument has to be of integer type and defines the hill spawning frequency, *i. e.*, every how many MD steps a new hill is added *(typically every 10 – 50 fs)*. The second argument is a real number and specifies the height of each new hill in units of kJ mol^{-1} *(typically 0.1 – 1.0 kJ mol^{-1})*. The third argument sets the width of the Gaussian hills σ (which is the standard deviation, **not** the variance σ^2) in "scale units" – see the `Scale` modifier above. In two-dimensional Metadynamics simulations, the width applies for both dimensions at the same time, and the scales of the two Colvars need to be adjusted to obtain the correct "aspect ratio" of the hill width. Standard choices for σ are 0.1 – 1.0 scale units. The hill spawning parameters can be changed at any point during a Metadynamics simulations, not modifying the hills which are already present. If spawning of new hills shall be temporarily suspended during a Metadynamics simulations, "`Metadynamics HillSpawn Off`" can be specified. If you want do delete all hills, consider the `Reset` modifier described below.

The `Store` modifier controls how often the current intermediate free energy profile is saved to disk. It expects one integer argument which specifies the number of MD simulation time steps between two such stores. In case of one-dimensional Metadynamics, these free energy profiles have the file names "`basename-metadynamics_profile_###.csv`", where "`###`" indicates the step number after which the profile was written. In addition to that, a file "`basename-metadynamics_profile_history.csv`" is written, which contains all the previously computed free energy profiles as columns, so that they can easily be printed in one single plot. For two-dimensional Metadynamics, Gnuplot source files with file names "`basename-metadynamics_2d_profile_###.gp`" are written, which can be converted into contour plots with the freeware tool Gnuplot *(runs both on Windows and GNU Linux)*. The raw data for these contour plots can be found in corresponding files named "`basename-metadynamics_2d_profile_###.gp.csv`". Note that in both cases, the free energy scale origin is set to the deepest free energy well, so that all numbers are positive. If the `Store` modifier is not specified, the free energy profiles are stored every 1000 MD steps per default.

The `WellTempered` modifier switches on well-tempered Metadynamics [808]. In contrast to standard Metadynamics, the free energy profile converges towards a limit for long runs with this approach. In short terms, this approach scales down the hill size at positions where already many hills have been spawned before, so that the changes in the bias potential become smaller over time *(convergence)*. The `WellTempered` modifier expects one real argument, which is the so-called bias temperature, specified in temperature units. The bias temperature should be chosen in a way so that $\frac{1}{2} \cdot k_\mathrm{B} \cdot T_\mathrm{Bias}$ is around the same size as the largest barrier which the simulation shall overcome. For example, a bias temperature of 12 000 K is well-suited to overcome barriers of around 100 kJ mol^{-1}. Note that the Metadynamics module needs to know the simulation temperature in order to reconstruct the free energy profile in a well-tempered Metadynamics run. Typically, a thermostat should be active during a Metadynamics run, keeping the simulation temperature constant. In this case, the temperature is simply obtained from the thermostat. However, if no thermostat for the region `all` is specified, the simulation temperature has to be specified manually for well-tempered Metadynamics. This can be achieved by the `temperature` modifier, which expects one real argument – the simulation temperature in temperature units.

The `Lagrange` modifier switches on extended Lagrangian Metadynamics [805]. In this variant, a virtual particle *(with mass and velocity)* moves in the space spanned by the Colvars, and the only connection between this particle and the real atomistic system is a harmonic spring. The bias potential *(the Gaussian hills)* only acts on the virtual particle. The first argument is the mass of the virtual particle in a.m.u. The second argument is the harmonic spring constant in units of kJ mol^{-1}, which is evaluated in scale units – see the `Scale` modifier above. Typical values depend on the system and Colvars, but might be 100 a.m.u. and 10 kJ mol^{-1}. Optionally, a third and fourth parameter can be given to switch on thermostating of the virtual particle. A simple Berendsen thermostat is applied here. The third argument is the target temperature of the virtual particle, and the fourth argument is the thermostat time constant τ. A good choice would be a target temperature of 100 K and $\tau = 10$ fs. Note that in contrast to the normal Berendsen thermostat, the virtual particle is only cooled, but never heated. In other words, the thermostat only becomes active if the instantaneous temperature of the virtual particle becomes larger than the target temperature. This is to ensure that the virtual particle can change its direction – otherwise, it might happen that it is driven in the same direction for very long time intervals.

The `Reset` modifier resets the bias profile, *i. e.*, it deletes all hills which had been spawned, so that the bias profile becomes flat again. All other parameters of the Metadynamics simulation are not modified. If, for example, hill spawning is still on, then new hills will be spawned in the next simulation run.

The `Off` modifier completely switches off Metadynamics. It deletes all hills and turns off the Metadynamics module. It also resets the choice of Colvars for Metadynamics, so you will need to use this first if you want to set up a second different Metadynamics run within the same input script. This modifier can only be given as first argument to the `Metadynamics` command, and no further arguments can follow.

A restart file for the Metadynamics module (file name "`basename.metarestart`") is written each time a new hill has been spawned. The *`Restart`* command detects this file and automatically restarts the Metadynamics run *(i. e., loads all hills and the positions and velocities of the extended Lagrangian virtual particle if active)*. However, this only happens when Metadynamics is active and set up at the time when the *`Restart`* command is invoked. The parameters for the Metadynamics simulation are **not** restarted. Therefore, leave all parameter settings via calls to the `Metadynamics` command in place in your input file, and simply call the *`Restart`* command after all those, directly before the *`Run`* command.

Please see also the discussion on Metadynamics in [**Section** *1.3*](#moldyn:sec_metadynamics).

This section is concluded with a full example for a two-dimensional well-tempered extended Lagrangian Metadynamics run with restart ability *(just run the same input again to continue the simulation where it ended last)*. The two Colvars are defined as distances between atoms. You need to adapt all parameters in blue to your question and system:

```
Timestep 0.5_fs
Initvel 350_K
Thermostat NHC 350_K Timecon 100.0_fs
Dump Position Stride 1 Filename "trajectory.xyz"
Manage_Colvar Define 1 Distance Atom 0 Atom 1
Manage_Colvar Define 2 Distance Atom 2 Atom 3
Metadynamics Colvar 1 Scale 1.0_A Wall Lower 3.0 50.0 Upper 10.0 50.0 Range 0.0 15.
↪0 100
Metadynamics Colvar 2 Scale 1.0_A Wall Lower 1.0 50.0 Upper 8.0 50.0 Range 0.0 13.
↪0 100
Metadynamics HillSpawn 40 0.5 0.5 Store 2000
Metadynamics WellTempered 6000_K
Metadynamics Lagrange 100.0 10.0 200.0_K 10.0_fs
Restart IfExists
Run 100000
```

Minimize

Mandatory Arguments:		—		
Optional Arguments:		*method*	Keyword	{ `Combined`, `LBFGS`, `Anneal` }
Modifiers:	`Steps`	*n*	Integer	—
	`MaxGrad`	*thres*	Real	[kJ mol^{-1} Å^{-1}]
	`RMSGrad`	*thres*	Real	[kJ mol^{-1} Å^{-1}]
	`TempConv`	*thres*	Real	[temperature]
	`Accel`	*value*	Real	—
	`Damp`	*value*	Real	—
	`StepLimit`	*value*	Real	[length]
	`History`	*n*	Integer	—
	`Noise`	*value*	Real	[length]
	`OnlyH`	—		

Performs a Cartesian energy minimization of the system. For molecules, this is less efficient than ORCA's built-in geometry optimization in internal coordinates *(i. e., requires more steps to converge)*. However, the algorithms employed here also work with large atom counts *(e. g., 50 000)* as sometimes encountered in QM/MM simulations, which is absolutely out of scope of ORCA's primary optimization module. Furthermore, the minimization also works under all types of constraints (which some limitations in the case of L-BFGS) that have been set with the `Constraint` command, and also includes the effect of the repulsive simulation cell if activated. Only atoms contained in the active region are displaced, while all other atoms are kept at their positions.

The simplest way of performing a minimization is simply calling the `Minimize` command without arguments. This defaults to the L-BFGS method, which is fairly robust and efficient. If the minimization seems unstable, try to reduce the `History` or `StepLimit` parameters. L-BFGS may sometimes show poor performance with constraints other than Cartesian type. Apart from that, there is also a simulated annealing method implemented, which can be selected by specifying `Anneal` as the first argument. In contrast to L-BFGS, the simulated annealing method works equally well with all types of constraints. There is also a `Combined` method, which is a combination of some L-BFGS steps in the beginning, followed by a simulated annealing run until the temperature falls below a threshold, and another final L-BFGS run until the convergence criteria are reached.

With the `Steps` modifier, the maximum number of minimization steps can be specified. If this number of steps has been performed, the minimization finishes, no matter if the convergence criteria are fulfilled or not. The default value is 500.

The `MaxGrad` and `RMSGrad` modifiers control the convergence thresholds for the largest gradient on some atom and the root mean square average of the gradients. The default values are currently set to 5.0 and 1.0 kJ mol^{-1} Å^{-1}, respectively, which is about the same criterion as the default setting in the primary ORCA geometry optimization.

If the `TempConv` modifier is given, a simulated annealing run finished after the temperature was monotonously decreasing within 5 successive steps, and dropped below the specified value. Note that the simulated annealing run will finish if either this condition is reached, or the gradient thresholds are observed. It is not required to fulfill both criteria.

The `Accel` modifier specifies the acceleration factor for simulated annealing runs *(has no effect on L-BFGS)*. As long as the angle between velocity vector and gradient vector of some atom is below 90 degrees, the gradient is multiplied by this factor and the velocity is multiplied by a fraction of this factor. This helps to enforce a faster movement in gradient direction. The default value is 4.0. If this feature is not desired, use `Accel 1.0` to switch it off (1.0 means "no artificial acceleration").

The `Damp` modifier is the damping factor for simulated annealing runs *(has no effect on L-BFGS)*. Atom velocities are multiplied by this factor in every integration step. The default value is 0.98. Smaller values will make the algorithm more stable and less prone to oscillations and overshoots, but will also require significantly more steps to converge. Don't use values ≥ 1, as then it won't be an "annealing" anymore :-)

The `StepLimit` modifier specifies the maximum displacement of any atom *(in length units)* that can happen in one step of a minimization run. This can help to avoid large, unreliable steps which could lead to abrupt jumps in geometry and very high potential energies. This modifier concerns both L-BFGS and simulated annealing runs.

Negative values disable the step limit. The step limit is disabled by default. If you need to switch it on, try something in the order of 0.1 Å.

The `History` modifier controls the depth of gradient and position vector history that is used in the L-BFGS method to approximate the inverse Hessian. The default value is 20. Smaller values can help to stabilize the algorithm.

With the `Noise` modifier, small random numbers can be added to the atom positions before the minimization starts. This can help to escape local maxima and saddle points in the minimization. For example, a minimization of an initially linear water molecule would not be able to leave this maximum – but with some random "noise", it will be possible. The modifier expects one real argument which specifies the maximum atom displacement in length units (something like 0.01 Å will be reasonable). This feature is switched off by default.

If the `OnlyH` modifier is given, all non-hydrogen atoms are removed from the active region before the minimization starts. After the minimization has finished, the original active region is restored. This is helpful if only hydrogen positions shall be optimized, *e. g.*, to refine experimental crystal structures.

PrintLevel

Mandatory Arguments:	*value*	Keyword	{ `Low`, `Medium`, `High`, `Debug` }
Optional Arguments:	—		
Modifiers: —			

Controls the amount of information which is printed to the screen during the simulation. `Debug` should be used only in rare cases, because it might slow the simulation down heavily.

The default value is `Medium`.

Randomize

Mandatory Arguments:	—		
Optional Arguments:	*seed*	Integer	—
Modifiers: —			

There are a few algorithms in the ORCA MD module which rely on random numbers, *e. g.*, the initialization of atom velocities with the *`Initvel`* command. These random numbers are so-called "pseudo-random numbers", produced by a deterministic generator. This generator has a *state*, which is simply an integer number. If initialized to the same state, the generator will always create the same sequence of "random" numbers. This sounds like a deficiency at first thought, but is a very important feature for scientific reproducibility and for debugging purposes. If you start the same MD input file with "random" velocity initialization a couple of times, the trajectory will be exactly identical in all runs.

However, there are cases in which this behavior is not desired, *e. g.*, if you want to average a property over multiple trajectories of the same system. In these cases, call the `Randomize` command in the beginning of the input. If no argument is given, the random number generator is initialized with the current system time as a seed. MD runs started at different times will have different random velocities in the beginning. If you want more control over this process, you can also specify a positive integer number as argument, which is used as initial random seed. Simulations started with the same seed argument will have identical initial random velocities (if all other system parameters such as atom count, atom types, … remain identical).

Without a call to `Randomize`, a seed of 1 is always used.

Restart

Mandatory Arguments:		—		
Optional Arguments:		*fname*	String	—
Modifiers:	IfExists	—		

Reads a restart file to continue a previous molecular dynamics run. Such a restart file is written after every simulation step, such that a crashed simulation may easily be recovered. The file name of the restart file may be given via *fname*; otherwise, it is deduced from the project's base name as `<basename>.mdrestart`.

If the `IfExists` modifier is specified, a restart is only performed if the restart file exists. The error and abort that would normally occur in case of a non-existent restart file are suppressed by this flag. This is useful in the first of a series of batch runs, where the restart file does not yet exist in the beginning.

Please note that the following quantities are stored to/loaded from restart files:

- Atom Positions
- Atom Velocities
- Thermostat internal state *(only for NHC)*
- Metadynamics hills and extended Lagrangian internal state
- Simulation step number and elapsed physical time

All other quantities (timestep, regions, thermostat, constraints, cells, etc.) are **not** restarted and need to be set in the input file, typically **before** the `Restart` command. It is safe to just call the `Restart` command immediately before the *`Run`* command.

Please see also the discussion on restarting simulations in [**Section** *1.3*](#moldyn:sec_restart).

Restraint

Mandatory Arguments:		*operation*	Keyword	{ `Add`, `Reset` }
			Keyword	`Colvar`
		colvar	Integer	—
Optional Arguments:		—		
Modifiers:	`Harmonic`	—		
	`Gaussian`	—		
	`Spring`	—		
	`Sigma`	*Gaussian Sigma*	Real	…
	`Height`	*Gaussian Height*	Real	kJ mol^{-1}
	`Target`	*target*	Real	…
	`Lower`	*lower wall*	Real	…
	`Upper`	*upper wall*	Real	…
	`Ramp`	*initial target*	Real	…
		final target	Real	…
	`Noprint`	—		

This command imposes restraints on collective variables ("Colvars") defined before via the *`Manage_Colvars`* command. As a first argument, it expects the kind of operation to perform, which can be `Add` and `Reset`. The second argument needs to be the keyword "`Colvar`", and the third argument is an integer number specifying the Colvar on which the operation shall be performed.

If the first argument is "`Add`", a new restraint is added to the specified Colvar. Note that an arbitrary number of restraints of different types can be active on a Colvar at the same time. The next argument after the Colvar number needs to be the type of the restraint. Currently, `Harmonic` and `Gaussian` are allowed. When adding harmonic restraints, the `Spring` modifier can be given, specifying the harmonic spring constant of the restraint in

kJ mol^{-1} unit^{-2}, where unit is the default physical unit of the Colvar (Angstrom for distances, degree for angles). If not specified, a value of 50 kJ mol^{-1} unit^{-1} is used. When adding Gaussian restraints, the `Height` and `Sigma` modifiers are allowed. The former sets the height of the Gaussian hill in kJ mol^{-1}, while the latter sets the width of the Gaussian function in physical Colvar units *(σ is the standard deviation, **not** the variance σ^2)*. The height can be either positive or negative, allowing for both Gaussian hills and Gaussian wells. If not specified, the default values of -10 kJ mol^{-1} for the height *(i. e., Gaussian well)* and 10 Colvar units *(e. g., Angstrom or degree)* for sigma are used.

The position of the new restraint is controlled via the `Target` modifier, which expects one real argument in Colvar units. If the restraint shall be an one-sided wall, the modifiers `Lower` and `Upper` can be used instead of `Target`. It is also possible to specify both `Lower` and `Upper` in order to define a lower and an upper wall at different positions in one command. If `Ramp` is given directly after `Target`, `Lower`, or `Upper`, a restraint with linearly moving target position over time is defined. `Ramp` expects two arguments, which are the initial restraint position, and the final restraint position after the next subsequent *`Run`* command.

The following example shows how to assign a harmonic two-sided restraint with different lower and upper wall parameters to a distance Colvar with number 7 that has been previously defined via the *`Manage_Colvars`* command:

```
Restraint Add Colvar 7 Harmonic Lower 400_pm Spring 50.0
Restraint Add Colvar 7 Harmonic Upper 800_pm Spring 80.0
```

By default, some additional data *(current position, potential energy, external force, internal force)* for each restraint is printed to a file with the name "`basename-restraints.csv`" in every MD step. This data can be used, *e. g.*, for thermodynamic integration. If a large number of restraints is defined, this can waste time and disk space. To switch this off for a restraint, specify the `Noprint` modifier when defining it.

If the first argument was "`Reset`", all restraints imposed on the specified Colvar are deleted. No further arguments or modifiers *(apart from the three mandatory arguments described above)* can be given.

Run

Mandatory Arguments:		*n*	Integer	—
Optional Arguments:		—		
Modifiers:	`StepLimit`	*value*	Real	[length]
	`CenterCOM`	—		

Performs a molecular dynamics run over n time steps with the current settings, applying the velocity Verlet algorithm to solve the equations of motion (see section *1.5.1*). You might want to call commands like *`Timestep`*, *`Initvel`*, *`Thermostat`*, and *`Dump`* before. Please note that only atoms within the `active` region will be subject to time integration. All other atoms will be skipped, and will therefore retain their initial positions.

The `StepLimit` modifier can be used to limit the maximum displacement of any atom in a MD time integration step. In addition to the displacement, also the velocities will be limited to a maximum of *value*$\cdot\Delta t$. This can help to stabilize the dynamics if the initial geometry is poor and large forces are acting (close atoms, etc.). The keyword expects one real argument in distance units. A reasonable choice would be 0.1 Å.

If the `CenterCOM` modifier is given, the center of mass (CoM) of the total system is kept fixed. Normally, the CoM should not drift anyway, because the velocity initialization is performed in a way which gives the CoM a zero initial velocity, and the conservation of momentum should keep it like that. However, numerical errors and massive *`Thermostats`* *(among other factors)* can break this momentum conservation, leading to a drift of the CoM over time. If this shall be avoided, specify this modifier.

If no call to *`Initvel`* occurred before this command, the atom velocities are initialized to zero. If no call to *`Timestep`* occurred before this command, a default time step of 0.5 fs is set.

You can cleanly end a MD run by creating an empty file with the name "`EXIT`" *(note the all-uppercase letters on case-sensitive file systems)*. On Unix operating systems such as GNU Linux, this can easily be achieved by the command "`touch EXIT`". will detect the file, abort the MD run, and delete the file. You will still get the remaining output *(such as the timing statistics)*, and you don't have to delete all the remaining "`.tmp`" files, which both would not be the case if you would have killed the process instead.

SCFLog

Mandatory Arguments:	*value*	Keyword	{ `Discard`, `Last`, `Append`, `Each` }
Optional Arguments:	—		
Modifiers: —			

Controls how/if the detailed output from the electron structure calculation (*i. e.*, integrals, scf, gradient, …) will be written to log files. `Discard` completely discards the output. `Last` only keeps the last output for each program call (useful to read error message if simulation aborts). `Append` redirects all the output into one single log file ("basename.scf.log", "basename.int.log", "basename.grad.log", …), appending each step at the end of the file. `Each` writes the output for each step and each program to different log files, which have the step number in their file names.

The amount of information which is printed to the SCF log file can be controlled by the standard ORCA print flags, such as "`%output PrintLevel Maxi end`". Note that by default, ORCA reduces the print level after the first SCF. Due to this, properties such as orbital energies and population analyses will only be printed once by default. If you want to keep the print level constant for subsequent SCF runs, disable this feature via "`%method ReducePrint false end`" in the ORCA input.

The default value is `Append`. Note that this can lead to large log files in long runs.

Screendump

Mandatory Arguments:	—
Optional Arguments:	—
Modifiers: —	

Prints the current state of the MD module (atom positions, velocities, potential and kinetic energy, cell properties, etc.) to the screen and log file in a well-defined and "grepable" format. This is mostly useful for unit testing, *e. g.*, to verify if the system state after a MD run equals the state obtained from some other ORCA binary distribution.

Thermostat

Mandatory Arguments:		*type*	Keyword	{ `Berendsen`, `CSVR`, `NHC`, `None` }
Optional Arguments:		*temperature*	Real	[temperature]
Modifiers:	`Timecon`	*tau*	Real	[time]
	`Ramp`	*target_temp*	Real	[temperature]
	`Chain`	*chain_length*	Integer	—
	`MTS`	*mts*	Integer	—
	`Yoshida`	*yoshida*	Integer	—
	`Region`	*region*	…	…
	`Massive`	—		

Changes the atom thermostat settings for subsequent simulation runs. "*Type*" sets the thermostat type. Currently, three thermostat types are implemented: Berendsen [815], Nosé–Hoover chains (NHC) [809, 810], and "Canonical Sampling through Velocity Rescaling" (CSVR) [811]. The very basic and robust Berendsen thermostat should only be used for early pre-equilibration runs, as it does **not** sample the canonical ensemble and leads to problems such as the flying ice cube effect. Both the NHC and the CSVR thermostats are very sophisticated, and correctly sample the canonical ensemble. One of these two should be used in all standard NVT simulations. Use `None` as type to disable the thermostat.

The optional *temperature* argument sets the target temperature to which the system is thermostated. If this argument is omitted, the temperature from the last call to the `Initvel` command is used (if no such call was invoked before, the simulation is aborted).

The `Timecon` modifier sets the coupling strength of the thermostat (large time constants correspond to weak coupling). The default value is 10 fs, which is a relatively strong coupling. For a production run, 100 fs would be appropriate. Values in the range of 10 . . . 100 fs are reasonable (see also section *1.5.3*).

If the `Ramp` modifier is used, a temperature ramp can be applied during a MD run. The final temperature at the end of the ramp has to be specified directly after the modifier. The initial temperature at the beginning of the ramp is taken from the *temperature* argument *(or from the last* `Initvel` *command if this argument is missing)*. The temperature ramp is applied only to the *`Run`* command which first follows the ramp definition. The slope of the ramp is chosen such that the final temperature is reached at the end of the run. Any subsequent *`Run`* command will simply use the final temperature for thermostating. To apply another temperature ramp, you need to explicitly define it again.

The `Chain`, `MTS`, and `Yoshida` modifiers only apply to NHC thermostats. They specify the chain length of the Nosé–Hoover chain (default: 3), the number of multiple time steps in which the thermostat integration is performed (default: 2), and the order of the Yoshida integrator used (default: 3, allowed: 1, 3, 5, 7), respectively. Normally, there is little need to modify one of these parameters. For more information, refer to the original publications [809, 810].

The `Massive` modifier activates *massive thermostating*, which means that each degree of freedom is assigned to an independent thermostat. This is useful for pre-equilibration runs (helps to reach energy equipartition) and should not be used during production runs, as it might heavily distort the dynamics. Note that massive thermostats also break the conservation of momentum (both linear and angular), so better specify the `CenterCOM` modifier for the *`run`* command if this is an issue. Please also note that massive NHC thermostats of large systems can be quite slow, because each NHC thermostat is a dynamical system on its own which needs to be time integrated.

With the `Region` modifier, the thermostat can be attached to a specific region (*i. e.*, subset of atoms). This modifier expects one argument, which is either the name of a pre-defined region or the number of a user-defined region *(see above)*. If not specified, the thermostat acts on the whole system. Multiple thermostats for multiple regions can be active at the same time, but each region can have only one attached thermostat at a time (re-defining will overwrite the thermostat settings).

The command "`Thermostat None`" will remove all thermostats from all regions. If you want to disable a thermostat for a specific region only, use "`Thermostat None Region r`", where `r` is the name or number of the region.

Please note that all three implemented thermostat types will show no effect (or unexpected effects) if the system's temperature is close to 0 K, as they all work by multiplying the velocities with a *(more or less complicated)* factor.

`Timestep`

Mandatory Arguments:	*dt*	Real	[time]
Optional Arguments:	—		
Modifiers:	—		

Sets the simulation time step Δt used to integrate the equations of motion for all following runs to dt. If your system contains hydrogen atoms, a time step not above 0.5 fs is recommended. If only heavier atoms are present, a larger time step may be chosen. A good estimate for a time step that still allows for an accurate simulation is $\Delta t = \sqrt{m} \cdot 0.5$fs, where m is the mass of the lightest atom in the system (in a.m.u.). This is one reason why some scientists perform simulations with fully deuterated compounds: It allows to increase the time step by a factor of ≈ 1.4 :-)

If this command is not invoked before a *`Run`* call, a default time step of 0.5 fs will be set before starting the run.

7.1.7 Scientific Background

In this section, some of the methods and algorithms used within ORCA's MD module are described in some more depth, with a focus on the scientific background.

Time Integration and Equations of Motion

The central concept of molecular dynamics simulations is to solve Newton's equations of motion (at least as long as the atom cores are treated classically). These read

$$\ddot{x}_i\left(t\right) = \frac{F_i\Big(\vec{x}\left(t\right)\Big)}{m_i}, \quad i = 1 \ldots N,$$

where $x_i\left(t\right)$ denotes the position of the i-th degree of freedom at time t, m the corresponding mass, and F_i the force acting upon this degree of freedom. As the force may depend on all positions, this is a coupled system of N ordinary differential equations (ODEs). In the general case, it is not possible to obtain an analytical solution of this system, and therefore numerical solution methods are applied. These are almost always based on discretizing the time variable and approximately solving the system by taking finite time steps.

Of all different methods to numerically solve coupled systems of ODEs, the *symplectic integration schemes* for Hamiltonian systems attained special attention in the field of molecular dynamics. They possess a very good conservation of energy. In contrast to many other methods, they show a reasonable behavior when investigating the long-term evolution of chaotic Hamiltonian systems (like, *e. g.*, MD simulations). Three popular such symplectic integration schemes are the *Leapfrog* algorithm, the *Verlet* method, and the *Velocity Verlet* integrator. Despite their different names, they are very similar. It can be easily seen that the Verlet and Velocity Verlet methods are algebraically equivalent (by eliminating the velocities from the Velocity Verlet algorithm), and it can be shown that, eventually, all three methods are identical.[3] All three methods are explicit integration methods with a global error of order 2, and therefore one order better than the semi-implicit Euler method, which is also a symplectic integration scheme. As the Velocity Verlet algorithm is the only of these three methods which yields velocities and positions at the same point in time, many popular molecular dynamics packages (CP2k, CPMD, LAMMPS) use this scheme. For the same reasons, the ORCA MD module uses the Velocity Verlet algorithm as time integration method.

The general equations of the Velocity Verlet scheme read

$$\begin{aligned}
\vec{x}\left(t+\Delta t\right) &= \vec{x}\left(t\right) + \vec{v}\left(t\right)\Delta t + \frac{1}{2}\vec{a}\left(t\right)\Delta t^2, \\
\vec{v}\left(t+\Delta t\right) &= \vec{v}\left(r\right) + \frac{\vec{a}\left(t\right) + \vec{a}\left(t+\Delta t\right)}{2}\Delta t.
\end{aligned}$$

By inserting

$$\vec{a}_i\left(t\right) = \frac{\vec{F}_i\left(t\right)}{m_i}, \quad i = 1 \ldots N,$$

one arrives at the two-step method

$$\begin{aligned}
\vec{x}_i\left(t+\Delta t\right) &= \vec{x}_i\left(t\right) + \vec{v}_i\left(t\right)\Delta t + \frac{\vec{F}_i\left(t\right)}{2m_i}\Delta t^2, && i = 1 \ldots N, \\
\vec{v}_i\left(t+\Delta t\right) &= \vec{v}_i\left(r\right) + \frac{\vec{F}_i\left(t\right) + \vec{F}_i\left(t+\Delta t\right)}{2m_i}\Delta t, && i = 1 \ldots N,
\end{aligned}$$

which is implemented in ORCA's MD module.

[3] Hairer, Lubich, Wanner, "Geometric Numerical Integration", Springer 2006.

Velocity Initialization

In the beginning of a MD simulation, it is often the case that only the initial positions of the atoms are known, but not the velocities. As MD simulations are performed at some finite temperature, it is a good idea to initialize the velocities in a way such that the desired simulation temperature is already present in the beginning. In statistical mechanics, it is often assumed that the velocity distribution of atoms is given by a Maxwell–Boltzmann distribution (which is strictly only the case in idealized gases). Therefore, it is a reasonable choice to initialize the atom's velocities according to the Maxwell–Boltzmann equation in the beginning of a MD simulation. The goal is to find an initial velocity distribution in which each degree of freedom possesses a similar amount of energy, such that the equipartition theorem is approximately fulfilled.

The scalar Maxwell–Boltzmann velocity distribution (leaving out the normalization factor) at temperature T is given by

$$f\left(v\right) = v^2 \exp\Big(-\frac{mv^2}{2k_BT}\Big).$$

To initialize the particle's velocities such that this distribution function is fulfilled, one starts with a series of normal-distributed random numbers with mean 0 and variance 1, denoted by $\mathcal{N}\left(0,1\right)$. The Cartesian velocity components for each atom are then computed by

$$v_{i,\alpha} := \sqrt{\frac{k_BT}{m_i}}\mathcal{N}\left(0,1\right), \quad \alpha \in \{x,y,z\}, \;\; i = 1\ldots N.$$

As the C++98 standard does not offer a platform-independent way of obtaining normal-distributed random numbers, these are internally computed from uniformly distributed random numbers by applying the *Box–Muller transform* [816]: Assuming that u_1 and u_2 are two uniformly distributed random numbers from the interval $[0,1]$, the equations

$$\begin{aligned} z_1 &:= \sqrt{-2\log\left(u_1\right)}\cos\left(2\pi u_2\right), \\ z_2 &:= \sqrt{-2\log\left(u_1\right)}\sin\left(2\pi u_2\right) \end{aligned}$$

yield two new random numbers z_1 and z_2 which obey a normal distribution with mean 0 and variance 1.

After the velocities have been initialized, the total linear momentum of the system will probably have some finite value other than zero. As the linear momentum is (approximately) conserved within a molecular dynamics simulation, this would result in the system drifting away into one direction during the course of the simulation, which is probably not desired. Therefore, the total momentum is explicitly set to zero after the Maxwell–Boltzmann initialization:

$$\begin{aligned} \vec{P}_{\text{tot}} &:= \sum_{i=1}^{N} m_i\vec{v}_{i,\text{old}}, \\ \vec{v}_{i,\text{new}} &:= \vec{v}_{i,\text{old}} - \frac{\vec{P}_{\text{tot}}}{m_iN}, \quad i = 1\ldots N. \end{aligned}$$

This, of course, might change the initial temperature. Therefore, a final step is performed, in which all velocity vectors are multiplied with a factor that is determined such that the initial temperature exactly matches the target value.

Thermostats

After the initial velocities have been initialized to some finite temperature, it might be assumed that one can simply start the time integration of the dynamical system (equivalent to the *NVE ensemble*), and the starting temperature would be approximately preserved. In a real system, however, there are (at least) two reasons why the temperature will strongly deviate from the initial value already after a few steps. First, the initial velocity distribution only considers the kinetic energy of the particles, but some amount of energy will be exchanged with the potential energy contribution (*e. g.*, bond stretching) immediately, altering the temperature. Secondly, the numerical errors introduced due to the finite time step (and in case of *ab initio* MD, also due to the approximate forces) will lead to a drift in energy and therefore in temperature. To counter these effects, it is often desirable to have a temperature control during the course of the simulation (which then runs in the *NVT ensemble*), which is called a thermostat.

There exist many different kinds of thermostats, ranging from simple expressions up to highly complex dynamical systems on their own. But all of them share a common issue: If the thermostat is coupled only weakly to the system, the temperature will change anyway. However, if the thermostat is coupled more strongly to the system (*i. e.*, intervenes stronger), then the dynamics of the simulation will change, no longer resembling the undisturbed original dynamics which one wants to investigate. Therefore, it is always a tradeoff between temperature stability and disturbed dynamics to decide how strong a thermostat should be coupled to the system.

In ORCA, currently three thermostats are implemented: The Berendsen thermostat [815], the Nosé–Hoover chain thermostat (NHC) [809, 810], and the "Canonical Sampling through Velocity Rescaling" thermostat (CSVR) [811].

Berendsen Thermostat

The Berendsen thermostat [815] is similar to the simple velocity rescaling scheme, but enhanced by a time constant τ to control the coupling strength. Let T_0 be the desired target temperature and T the current temperature of the system. Then the temperature gradient caused by the thermostat can be expressed as

$$\frac{dT}{dt} = \frac{T_0 - T}{\tau}.$$

Considering the fact that discrete time steps Δt are used, the correction factor for the velocities in each time step is determined by

$$f := \sqrt{1 + \frac{\Delta t\,(T_0 - T)}{T\tau}}$$

The new velocities are then easily obtained as

$$\vec{v}_{i,\text{new}} := f \cdot \vec{v}_{i,\text{old}}, \quad i = 1 \ldots N.$$

Let's consider some special cases. If $\tau = \Delta t$, the whole temperature deviation from T_0 is corrected immediately, such that the temperature is always exactly kept at the target value. This is identical to simple velocity rescaling (without any time constant), which is known to work poorly for most systems (a single harmonic oscillator would, *e. g.*, simply explode). With a larger time constant $\tau > \Delta T$, the coupling strength is reduced, leading to reasonable results. Typically, a value of τ in the range of $20 \ldots 200 \cdot \Delta T$ will be applied. For $\tau \to \infty$, the coupling strength goes to zero, such that the thermostat is no longer active. Values of $\tau < \Delta T$ are not allowed.

From the formula, it becomes clear that a Berendsen thermostat will have no effect if the system has a temperature of 0 K (or in the "massive" case: if the considered degree of freedom has 0 K), because it is based on multiplying the velocities by a factor to modify the temperature. Therefore, this type of thermostat can't be used to heat a system up starting from 0 K.

Constraints

Unlike restraints, constraints are geometric relations which are strictly enforced at every time (*i. e.*, they do not fluctuate around their target value). Many molecular dynamics techniques make use of geometric constraints (*e. g.*, to keep water molecules rigid, or to fix some reaction coordinate). Standard BOMD describes the nuclei as point charges in space, such that the motion of the atoms is governed by the laws of classical mechanics. Systems in classical mechanics can be described by the Lagrange formalism, which contains a well established sub-formalism for holonomic constraints, namely the method of Lagrange multipliers.

However, molecular dynamics discretizes time to solve the equations of motions with finite time steps, often using a Verlet integrator. With discretized time, it is slightly more involved to enforce and keep exact constraints. Within the last decades, algorithms have been developed to do so. One famous among them is the SHAKE algorithm. However, it comes with the disadvantage of only enforcing the constraints in the positions, not in the velocities. This may lead to problems such as artificially high temperature values due to "hidden" velocities along the constrained directions. An extension of SHAKE which also enforces the constraints for the velocities is the RATTLE algorithm, which is implemented in the AIMD module of ORCA.

The RATTLE scheme is a generalization of the Velocity Verlet integrator to allow for constraints. This means that RATTLE is not applied in addition to the Velocity Verlet integrator, but replaces it. In case of no active constraints,

both methods are identical. A system of coupled constraints cannot be solved exactly in one step, and RATTLE uses an iterative approach to enforce all constraints simultaneously. This is often a matter of concern with respect to performance. However, in AIMD, the energy and gradient calculations typically take seconds or even minutes per step, such that the additional computation time for iteratively solving the constraints can be totally neglected.

As an iterative procedure, RATTLE is not able to give exact solutions, but only converged up to a given tolerance. In the ORCA MD module, the tolerance is currently set to 10^{-2} pm for distances, and 10^{-4} degree for angles and dihedral angles. This tolerance is typically reached within a few dozen iterations. In some cases, it might happen that the RATTLE iterations do not converge to the required tolerance. This is typically the case if the set of constraints is over-determined or contradictory.

The mathematical and technical details of RATTLE are not described here, they can be found in the literature. The general concept of RATTLE was suggested by Andersen [817]. The original article only covered distance constraints. A follow-up work describes how to handle any holonomic constraints, in particular how to constrain angles and dihedral angles [818]. The Wilson vectors (*i. e.*, derivatives of angles and dihedral angles with respect to Cartesian atom positions) are taken from Wilson's original work [819].

CHAPTER

EIGHT

WORKFLOWS AND AUTOMATIZATION

8.1 ORCA Python Interface (OPI)

The ORCA Python interface (OPI) is a versatile tool that provides seamless access to ORCA calculations through Python. The OPI library offers many functions and classes to easily define calculation settings, perform different kinds of ORCA calculations, and process the resulting output data. It is open-source and freely available at GitHub. For detailed installation instructions and usage examples, please refer to the official OPI documentation.

Added in version 6.1.0: The first version of OPI was released with ORCA 6.1. Please note that OPI's open-source development is managed independently of the ORCA software itself. For the latest updates and changes, consult the OPI GitHub repository and the OPI documentation.

8.1.1 Installation

OPI can either be installed directly from PyPI:

```
pip install orca-pi
```

or from GitHub:

```
git clone https://github.com/faccts/opi.git
cd opi
python3 -m venv .venv
source .venv/bin/activate
python3 -m pip install .
```

In order to use OPI properly we recommend to install it in a venv environment. For more instructions refer to the OPI documentation.

8.2 Compound

Compound is a form of sophisticated scripting language that can be used directly in the input of ORCA. Using '*Compound*' the user can combine various parts of a normal ORCA calculation to evaluate custom functions of his own. In order to explain its usage, we will use an example. For a more detailed description of this module the user is referred to section *More Details on Compound*.

8.2.1 example

As a typical example we will use the constrained optimization describing the "umbrella effect" of NH_3. The script will perform a series of calculations and in the end it will print the potential of the movement plus it will identify the minima and the maximum. The corresponding compound script is the one shown below.

```
# ---------------------------------------------
# Umbrella coordinate mapping for NH3
# Author: Frank Neese
# ---------------------------------------------
variable JobName = "NH3-umbrella";
variable amin    = 50.0;
variable amax    = 130.0;
variable nsteps  = 21;
Variable energies[21];

Variable angle;
Variable JobStep;
Variable JobStep_m;
variable step;

Variable method = "BP86";
Variable basis  = "def2-SVP def2/J";

step  = 1.0*(amax-amin)/(nsteps-1);

# Loop over the number of steps
# ----------------------------
for iang from 0 to nsteps-1 do
  angle    = amin + iang*step;
  JobStep  = iang+1;
  JobStep_m= JobStep-1;
  if (iang>0) then
    Read_Geom(JobStep_m);
    New_step
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom constraints
        {A 1 0 2 &{angle} C}
        {A 1 0 3 &{angle} C}
        {A 1 0 4 &{angle} C}
        end
      end

    Step_End
  else
    New_step
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom constraints
        {A 1 0 2 &{angle} C}
        {A 1 0 3 &{angle} C}
        {A 1 0 4 &{angle} C}
        end
      end

      * int 0 1
      N 0 0 0 0.0 0.0 0.0
      DA 1 0 0 2.0 0.0 0.0
      H 1 2 0 1.06 &{angle} 0.0
      H 1 2 3 1.06 &{angle} 120.0
      H 1 2 3 1.06 &{angle} 240.0
```

(continues on next page)

(continued from previous page)

```
      *
    Step_End
  endif
   Read energies[iang] = SCF_ENERGY[jobStep];
   print(" index: %3d Angle %6.2lf Energy: %16.12lf Eh\n", iang, angle,␣
↪energies[iang]);
EndFor

# Print a summary at the end of the calculation
# ---------------------------------------------
print("//////////////////////////////////////////////////////////\n");
print("// POTENTIAL ENERGY RESULT\n");
print("//////////////////////////////////////////////////////////\n");
variable minimum,maximum;
variable Em,E0,Ep;
variable i0,im,ip;
for iang from 0 to nsteps-1 do
  angle   = amin + 1.0*iang*step;
  JobStep = iang+1;
  minimum = 0;
  maximum = 0;
  i0      = iang;
  im      = iang-1;
  ip      = iang+1;
  E0      = energies[i0];
  Em      = E0;
  Ep      = E0;
  if (iang>0 and iang<nsteps-1) then
    Em = energies[im];
    Ep = energies[ip];
  endif
  if (E0<Em and E0<Ep) then minimum=1; endif
  if (E0>Em and E0>Ep) then maximum=1; endif
  if (minimum = 1 ) then
    print(" %3d  %6.2lf %16.12lf (-)\n",JobStep,angle, E0 );
  endif
  if (maximum = 1 ) then
    print(" %3d  %6.2lf %16.12lf (+)\n",JobStep,angle, E0 );
  endif
  if (minimum=0 and maximum=0) then
    print(" %3d  %6.2lf %16.12lf    \n",JobStep,angle, E0 );
  endif
endfor
print("//////////////////////////////////////////////////////////\n");

End # end of compound block
```

Let's start with how somebody can execute this input. In order to run it, the easiest way is to save it in a normal text file, using the name "umbrella.cmp" and then use the following ORCA input file:

```
%Compound "umbrella.cmp"
```

nothing more is needed. ORCA will read the compound file and act appropriately.

A few notes about this ORCA input. First, there is no simple input line, (starting with *"!"*). A simple input is not required when one uses the *Compound* feature, but In case the user adds a simple input, all the information from the simple input will be passed to the actual compound jobs.

In addition, if one does not want to create a separate compound text file, it is perfecly possible in ORCA to use the compound feature as any other ORCA block. This means that after the *%Compound* directive, instead of giving the filename one can append the contents of the Compound file.

As we will see, inside the compound script file each compound job can contain all information of a normal ORCA

input file. There are two very important exceptions here: The number of processors and the *MaxCore*. These information should be set in the initial ORCA input file and not in the actual compound files.

The Compound block has the same structure like all ORCA blocks. It starts with a *"%"* and ends with *"End"*, if the input is not read from a file. In case the compound directives are in a file, as in the example above, then simply the filename inside brackets is needed and no final *END*.

8.2.2 Defining variables

As we pointed out already, it is possible to either give all the information for the calculations and the manipulation of the data inside the Compound block or create a normal text file with all the details and let ORCA read it. The latter option has the advantage that one can use the same file with more than one geometries. In the previous example we refer ORCA to an external file. The file *"umbrella.cmp"*, that contains all necessary information.

Let's try to analyse now the Compound *"umbrella.cmp"* file.

```
# ---------------------------------------------
# Umbrella coordinate mapping for NH3
# Author: Frank Neese
# ---------------------------------------------
variable JobName = "NH3-umbrella";
variable amin    = 50.0;
variable amax    = 130.0;
variable nsteps  = 21;
Variable energies[21];

Variable angle;
Variable JobStep;
Variable JobStep_m;
variable step;

Variable method = "BP86";
Variable basis  = "def2-SVP def2/J";

step  = 1.0*(amax-amin)/(nsteps-1);
```

The first part contains some general comments and variable definitions. For the comments we use the same syntax as in the normal ORCA input, through the *"#"* symbol. Plase not that more than one *"#"* symbols in the same line cause an error.

After the initial comments we see some declarations and definitions. There are many different ways to declare variables described in detail in section *Variables - General*.

All variable declarations begin with the directive *Variable* which is a sign for the program to expect the declaration of one or more new variables. Then there are many options, including defining more than one variable, assigning also a value to the variable or using a list of values. Nevertheless all declarations **MUST** finish with the *;* symbol. This symbol is a message to the program that this is the end of the current command. The need of the *;* symbol in the end of each command is a general requirement in *Compound* and there are only very few exceptions to it.

8.2.3 Running calculations

```
# Loop over the number of steps
# ----------------------------
for iang from 0 to nsteps-1 do
  angle    = amin + iang*step;
  JobStep  = iang+1;
  JobStep_m= JobStep-1;
  if (iang>0) then
    Read_Geom(JobStep_m);
    New_step
```

(continues on next page)

(continued from previous page)

```
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom
        constraints
          {A 1 0 2 &{angle} C}
          {A 1 0 3 &{angle} C}
          {A 1 0 4 &{angle} C}
        end
      end
    Step_End
  else
    New_step
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom
        constraints
          {A 1 0 2 &{angle} C}
          {A 1 0 3 &{angle} C}
          {A 1 0 4 &{angle} C}
        end
      end
      * int 0 1
        N 0 0 0 0.0 0.0 0.0
        DA 1 0 0 2.0 0.0 0.0
        H 1 2 0 1.06 &{angle} 0.0
        H 1 2 3 1.06 &{angle} 120.0
        H 1 2 3 1.06 &{angle} 240.0
      *
    Step_End
  endif
  Read energies[iang] = SCF_ENERGY[jobStep];
  print(" index: %3d Angle %6.2lf Energy: %16.12lf Eh\textbackslash{}n", iang,␣
↪angle, energies[iang]);
EndFor
```

Then we have the most information dense part. We start with the definition of a *for* loop. The syntax in compound for *for* loops is:

For *variable* **From** *startValue* **To** *endValue* **Do**
directives
EndFor

As we can see in the example above, the *startValue* and *endValue* can be constant numbers or previously defined variables, or even functions of these variables. Keep in mind that they have to be integers. The signal that the loop has reached it's end is the *EndFor* directive. For more details with regard to the *for* loops please refer to section *For*.

Then we proceed to assign some variables.

```
angle     = amin + iang*step;
JobStep   = iang+1;
JobStep_m = JobStep-1;
```

The syntax of the variable assignement is like in every programming language with a variable, followed with the = symbol and then the value or an equation. Please keep in mind, that the assignement **must** always finish with the *;* symbol.

The next step is another significant part of every programming language, namely the *if* block. The syntax of the *if* block is the following:

if (*expression to evaluate*) **Then**
directives
else if (*expression to evaluate*) **Then**
directives

else
directives
EndIf

The *else if* and *else* parts of the block are optional but the final *EndIf* must always signal the end of the *if* block. For more details concerning the usage of the *if* block please refer to section *If* of the manual.

Next we have a command which is specific for compound and not a part of a normal programming language. This is the *ReadGeom* command. It's syntax is:

Read_Geom(*integer value*);

Before explaining this command we will proceed with the next one in the compound script and return for this one.

The next command is the basis of all compound scripts. This is the *New_Step* Command. This command signals compound that a normal ORCA calculation follows. It's syntax is:

New_Command *Normal ORCA input* **Step_End**

Some comments about the *New_Step* command. Firstly, inside the *New_Step - Step_End* commands one can add all possilbe commands that a normal ORCA input accepts. We should remember here that the commands that define the number of processors and the *MaxCore* command will be ignored.

A second point to keep in mind is the idea of the *step*. Every *New_Step - Step_End* structure corresponds to a step, starting counting from 1 (The first ORCA calculation). This helps us define the property file that this calculation will create, so that we can use it to retrieve information from it.

A singificant feature in the *New_Step - Step_End* block. is the usage of the structure **&{*variable*}** . This structure allows the user to use variables that are defined outside the *New_Step - Step_End* block inside it, making the ORCA input more generic. For example, in the script given above, we build the *basename* of the calculations

```
%base "&{JobName}.step&{JobStep}"
```

using the defined variables *JobName* and *JobStep*. For more details regarding the usage of the **&{}** structure please refer to section *&* while for the *New_Step - Step_End* structure please refer to the section *NewStep*.

Finally, a few comments about the geometries of the calculation. There are 3 ways to provide a geometry to a *New_Step - Step_End* calculation. The first one is the traditional ORCA input way, where we can give the coordinates or the name of a file with coordinates, like we do in all ORCA inputs. In *Compound* though, if we do not pass any information concerning the geometry of the calculation, then *Compound* will automatically try to read the geometry of the previous calculation. This is the second (implicit) way to give a geometry to a compound Step. Then there is a third way and this is the one we used in the example above. This is the **Read_Geom** command. The syntat of this command is:
Read_Geom (*Step number*);
We can use this command when we want to pass a specific geometry to a calculation that is not explicitly given inside the *New_Step - Step_End* structure and it is also not the one from the previous step. Then we just pass the number of the step of the calculation we are interesting in just before we run our new calculation. For more details regarding the *Read_Geom* command please refer to section *Read_Geom*.

8.2.4 Data manipulation

One of the most powerfull features of *Compound* is it's direct access to properties of the calculation. In order to use these properties we defined the *Read* command. In the previous example we use it to read the SCF energy of the calculation:

```
Read energies[iang] = SCF\_ENERGY[jobStep];
```

The syntax of the command is:

Read *variable name* **=** *property*

where *variable name* is the name of a variable that is already defined, *property* is the property from the known properties found in table *List of known Properties* and *step* is the step of the cal-

culation we are interested in. For more details in the *Read* command please refer to section sec:workflowsautomatization.compound_detailed.commands.propertyFile.read.

```
# Print a summary at the end of the calculation
  # ---------------------------------------------
  print("//////////////////////////////////////////////////////////\\n");
  print("// POTENTIAL ENERGY RESULT\\n");
  print("//////////////////////////////////////////////////////////\\n");
  variable minimum,maximum;
  variable Em,E0,Ep;
  variable i0,im,ip;
  for iang from 0 to nsteps-1 do
    angle   = amin + 1.0*iang*step;
    JobStep = iang+1;
    minimum = 0;
    maximum = 0;
    i0      = iang;
    im      = iang-1;
    ip      = iang+1;
    E0      = energies[i0];
    Em      = E0;
    Ep      = E0;
    if (iang>0 and iang<nsteps-1) then
      Em = energies[im];
      Ep = energies[ip];
    endif
    if (E0<Em and E0<Ep) then minimum=1; endif
    if (E0>Em and E0>Ep) then maximum=1; endif
    if (minimum = 1 ) then
      print(" %3d  %6.2lf %16.12lf (-)\textbackslash{}n",JobStep,angle, E0 );
    endif
    if (maximum = 1 ) then
      print(" %3d  %6.2lf %16.12lf (+)\textbackslash{}n",JobStep,angle, E0 );
    endif
    if (minimum=0 and maximum=0) then
      print(" %3d  %6.2lf %16.12lf    \textbackslash{}n",JobStep,angle, E0 );
    endif
  endfor
  print("//////////////////////////////////////////////////////////\\n");
```

Once all data are available we can use them in equations like in any programming language.

The syntax of the print statement is:

print(*format string, [variables]***);**

For example in the previous script we use it like:

```
print(" %3d  %6.2lf %16.12lf  \n",JobStep,angle, E0 );
```

where *%3d, %6.2lf* and *%16.2lf* are format identifiers and *JobStep, angle* and *E0* are previously defined variables. The syntax follows closely the widely accepted syntax of the *printf* command in the programming language C. For more details regarding the *print* statememnt please refer to section: *Print*.

Similar to the *print* command are the *write2file* and *write2string* commands that are used to write instead of the output file, either to a file we choose or to produce a new string.

Finally it is really importnat not to forget that every compound file should finish with a final **End**.

Once we run the previous example we get the following output:

```
//////////////////////////////////////////////////////////
// POTENTIAL ENERGY RESULT
//////////////////////////////////////////////////////////
```

(continues on next page)

(continued from previous page)

```
1   50.00 -56.486626696200
2   54.00 -56.498074637200
3   58.00 -56.505200120800
4   62.00 -56.508823168800
5   66.00 -56.509732863600 (-)
6   70.00 -56.508724734300
7   74.00 -56.506590613800
8   78.00 -56.504070086000
9   82.00 -56.501791816800
10   86.00 -56.500229017900
11   90.00 -56.499674856600 (+)
12   94.00 -56.500229018100
13   98.00 -56.501791817200
14  102.00 -56.504070082800
15  106.00 -56.506590613300
16  110.00 -56.508724733100
17  114.00 -56.509732863700 (-)
18  118.00 -56.508823172900
19  122.00 -56.505200132200
20  126.00 -56.498074642900
21  130.00 -56.486626729200
//////////////////////////////////////////////////////
```

with the step, the angle for the corresponding step, the energy of the constrained optimized energy plus the symbols for the two minima and the maximum in the potential.

8.3 More Details on Compound

In this part we describe "Compound" in more detail.

8.3.1 Commands

Below is a list of all available commands available in *Compound*

———————————— Dataset Related ————————————

- Dataset(*Dataset*)
- MakeReferenceFromDir(*D.MakeReferenceFromDir*)
- Print(*D.Print*)

———————————— File Handling Related ————————————

- CloseFile (*CloseFile*)
- OpenFile (*OpenFile*)

————————————- *For* Loop Related ————————————-

- Break (*Break*)
- Continue (*Continue*)
- EndFor (*EndFor*)
- For (*For*)

———————————— Geometry Related ————————————-

- Geometry(*Geometry*)
- BohrToAngs(*G.BohrToAngs*)

- CreateBSSE(*G.CreateBSSE*)
- CreateFragments(*G.CreateFragments*)
- DisplaceAtom(*G.DisplaceAtom*)
- FollowNormalMode(*G.FollowNormalMode*)
- GetAngle(*G.GetAngle*)
- GetAtomicNumbers(*G.GetAtomicNumbers*)
- GetBondDistance(*G.GetBondDistance*)
- GetCartesians(*G.GetCartesians*)
- GetGhostAtoms(*G.GetGhostAtoms*)
- GetNumOfAtoms(*G.GetNumOfAtoms*)
- MoveAtomToCenter(*G.MoveAtomToCenter*)
- Read(*G.Read*)
- RemoveAtoms(*G.RemoveAtoms*)
- RemoveElements(*G.RemoveElements*)
- WriteXYZFile(*G.WriteXYZFile*)

——————————— GOAT Related ———————————-

- GOAT(*GOAT*)
- Get_Energy (*Goat.Get_Energy*)
- Get_Num_Of_Geometries (*Goat.Get_Num_Of_Geometries*)
- ParseEensembleFfile (*Goat.ParseEnsembleFile*)
- Set_Basename (*Goat.Set_Basename*)
- Print (*Goat.Print*)
- WriteXYZFile (*Goat.WriteXYZFile*)

——————————— *If* block Related ———————————

- If (*If*)

———————————- Linear Algebra Related ———————————-

- Diagonalize(*Diagonalize*)
- InvertMatrix(*InvertMatrix*)
- Mat_p_Mat(*Mat_p_Mat*)
- Mat_x_Mat(*Mat_x_Mat*)
- Mat_x_Scal(*Mat_x_Scal*)

——————————— ORCA calculation Related ———————————

- Basenames(*Basenames*)
- ReadMOs(*ReadMOs*)

——————————— Program flow Related ———————————

- Abort (*Abort*)
- End (*End*)
- EndRun (*EndRun*)
- GoTo (*GoTo*)

——————— Property File Related ———————-

- GetNumOfInstances(*GetNumOfInstances*)
- ReadProperty(*ReadProperty*)

——————— String Handling Related ———————

- GetBasename (*S.GetBasename*)
- GetChar (*S.GetChar*)
- GetSuffix (*S.GetSuffix*)
- Print (*Print*)
- Write2File (*Write2File*)
- Write2String (*Write2String*)

——————————— Step Related ———————————

- & (*&*)
- Alias (*Alias*)

——————— Timer Related ———————————

- Timer (*Timer*)
- Last (*T.Last*)
- Reset (*T.Reset*)
- Start (*T.Start*)
- Stop (*T.Stop*)
- Total (*T.Total*)

——————— Variables Related ———————————-

- Variables (*Variables - General*)
- Variables - Assignment (*Variables - Assignment*)
- Variables - Declaration (*Variables - Declaration*)
- Variables - Functions (*Variables - Functions*)
- Variable - With (*With*)
- GetBool() *V.GetBool()*
- GetDim1() *V.GetDim1()*
- GetDim2() *V.GetDim2()*
- GetDouble() *V.GetDouble()*
- GetInteger() *V.GetInteger()*
- GetSize() *V.GetSize()*
- GetString() *V.GetString()*
- PrintMatrix() *V.PrintMatrix()*
- Write2File (*Write2File*)
- Write2String (*Write2String*)

&

The & symbol has a special meaning in the compound block. Using this symbol inside the *NewStep* - *StepEnd* block the user can use variables that are defined outside the block. Both string and numerical variables are allowed.

Syntax:

&{*variable*}

Example

```
# ------------------------------------------
# This script checks the options for the
#             '&' symbol
# ------------------------------------------
%Compound

  Variable method   = "BP86";              #string variable'
  Variable basis    = "def2-TZVP def2/J";
  Variable name     = "base";
  Variable number   = 0;                   #integer variable
  Variable distance = 0.8;                 #double variable
  New_step
    ! &{method} &{basis} TightSCF
    %base "&{name}_&{number}"              #combination of variables
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 &{distance}
    *
  Step_End
  # ------------------------------------------
  # Add some printing
  # ------------------------------------------
  print("SUMMARY OF VARIABLES\n");
  print("Method:   %s\n",    method);
  print("Basis:    %s\n",    basis);
  print("Name:     %s\n",    name);
  print("Number:   %d\n",    number);
  print("Distance: %.2lf\n", distance);
End
```

Abort

Abort is used when the user wants to exit the program instantly. **Syntax:**
Abort;
or alternatively:
Abort

Example:

```
%Compound
for i from 0 to 4 do
  print("i: %d\n", i);
  if (i=2) then
    abort;
  endif
endfor
End
```

Alias

Alias is used to replace an integer number with a more representative string. It is useful when one performs more than one calculations and the step numbers become too complicated to evaluate. In this case using Alias_Step after the Step_End command will connect the preceeding calculation step number with the provided name.

Syntax:
Alias *name*;
or alternatively:
Alias *name*
Example:

```
# ----------------------------------
# This script checks 'alias' keyword
# ----------------------------------
Variable numOfSteps = 20;
Variable Range      = 4.0;
Variable distStart  = 0.4;
Variable Step       = Range/numOfSteps;
Variable distance;
Variable Energies[numOfSteps];
Variable simpleInput  = "BP86 def2-SVP def2/J";

For index from 0 to numOfSteps-1 Do
  Distance =  distStart+ index*Step;
  New_Step
    !&{simpleInput}
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 &{Distance}
    *
  Step_End
  Alias currStep;
  print("Current step: %d\n", currStep);
  Read Energies[index] = JOB_INFO_TOTAL_EN[currStep];
EndFor

print("--------------------------------------\n");
print("       Compound Printing              \n");
print("%s  %12s  %16s \n","Step",  "Distance", "Energy");
print("--------------------------------------\n");
For index from 0 to numOfSteps - 1 Do
  Distance = distStart + index*Step;
  print("%4d  %12.4lf   %16.8lf \n", index, Distance, Energies[index]);
EndFor

End
```

Note

For the **Alias** command the final ';' is optional.

Basenames

Basenames in a variable that is automatically created in *Compound* everytime a *NewStep* is used and it holds the name of this step. It can be used to recover this name during running of a calculation. It practically is a vector were value zero corresponds to the main calculation before any *new_step* command.

Example:

```
*xyz 0 1
  H 0.0 0.0 0.0
  H 0.0 0.0 0.8
*
%Compound
  new_step
    !BP86
  step_end
  alias lala;
  print("Basaenames[0]: %s\n", basenames[0]);
  print("Basaenames[1]: %s\n", basenames[lala]);
End
```

Break

Break can be used inside a *For* loop (see (see *For*)) when one needs to break the loop under certain conditions. The syntax is the following:

Syntax:
For variable From Start value To End value Do
commands
break ;
commands
EndFor;

NOTE both versions *break* and *break;* are legal.

What break actually does is to set the running index of the loop to the last allowed value and then jump to the *EndFor* (see *EndFor*)).

Example:

```
#
# This a script to check Compound 'break' command
#
%Compound
print(" Test for 'break'\n");
print(" It should print 0, 1 and 2\n");
for i from 0 to 6 Do
  if (2*i > 4) then
    break
  endIF
  print("index: %d\n", i);
EndFor

print("Continued outside the 'for' loop\n");

End
```

CloseFile

When a file is opened in *Compound* using the *openFile* command (see *OpenFile*), then it must be closed using the *closeFile* command.

Syntax:
closeFile(file);

file is the file pointer created from the *openFile* command. For an example see paragraph *OpenFile*.

Continue

Continue can be used inside a *For* loop (see (see *For*)) when one needs to skip the current step of the loop and proceed to the next one. The syntax is the following:

Syntax:
For variable From Start value To End value Do
commands
continue ;
commands
EndFor;

NOTE both versions *continue* and *continue;* are legal.

What continue actually does is to jump to the *EndFor* (see *EndFor*)).

Example:

```
# ----------------------------------------------------
# This is a script to check Compound 'continue' command
# ----------------------------------------------------
%Compound
print(" Test for 'continue'\n");
print(" It should print 0, 1, 2 and 4\n");
for i from 0 to 4 Do
  if ( i=3) then
    continue;
  endIf
  print("index: %d\n", i);
EndFor
End
```

Dataset

In *Compound* we have *Dataset* objects. These objects can be treated like normal variables of type '*compDataset*'. An important difference between normal variables and *Dataset* variables is the declaration. Instead of the normal:

```
Variable x;
```

we excplicitly have to declare that this is a dataset. So the syntax for a dataset declaration is:

Syntax:

```
Dataset mySet;
```

NOTE in the case of datasets we do not allow multiple dataset declarations per line.

Below is a list of functions that work on *Dataset*.

- MakeReferenceFromDir(*D.MakeReferenceFromDir*)
- Print(*D.Print*)

Example:

```
# ----------------------------------------------------
# This is an example script for dataset definition
# ----------------------------------------------------
%Compound
  dataset mySet;
  mySet.Print();
End
```

D.MakeReferenceFromDir

MakeReferenceFromDir command acts on a dataset object (see *Dataset*). It creates a json reference file based on the *xyz files of the current folder. **NOTE** By default all charges and multplicities will be set to 0 and 1 respectively.

Syntax:
mySet.MakeReferenceFromDir(dirName);

Where:

- *mySet* is a dataset object that is already declared
- *dirName* The name of a directory that should contain some xyz files.

Example:

```
%Compound
  # ------------------------------------
  #   This is a compound script that will
  #   check dataset.MakeReferenceFromDir
  #   function
  #   NOTE: The script assumes that some
  #         xyz files rest in the current
  #         directory
  # ------------------------------------

  # First some definitions
  Variable name = "mySet";
  Variable numOfMolecules = 0;
  dataset mySet;
  Variable myDir="./";
  mySet.MakeReferenceFromDir(myDir);
  mySet.ReadReferenceFile();
  mySet.Print();
End
```

D.Print

Print command acts on a dataset object (see *Dataset*). It prints all details of the specific dataset object.

Syntax:
mySet.Print();

Where:

- *mySet* is a dataset object that is already declared

Example:

```
# ----------------------------------------------------
# This is an example script for dataset definition
# ----------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
%Compound
  dataset mySet;
  mySet.Print();
End
```

Diagonalize

Compound can peform matrix algebraic operations, one of the available algebraic operation is matrix diagonalization. Be carefull that the matrix, that is to be diagonalized, **must be** a square symmetric matrix. It is also important to remember that only the upper triangle part of the matrix will be used for the diagonalization. If everything proceeds smoothly then the function will return the eigenvectors and eigenvalues of the matrix.

Syntax:
A.Diagonalize(eigenValues, eigenVectors);

Where:

- *A:* The matrix to be diagonalized.
- *eigenValues:* The vector with the eigenvalues
- *eigenVectors:* The square matrix with the eigenvectros of the initial matrix.

Example:

```
# ----------------------------------------------------
# This is an example script for diagonalization
# ----------------------------------------------------
%Compound
  Variable Dim=3;
  Variable A[Dim][Dim];
  Variable eigenVal;
  Variable eigenVec;
  for i from 0 to Dim-1 Do
    for j from 0 to Dim-1 Do
      if (i<=j) then
        A[i][j] = i+j+1;
      else
        A[i][j] =  A[j][i];
      EndIf
    EndFor
  EndFor
  A.Diagonalize(eigenVal, eigenVec);
  A.PrintMatrix();
  eigenVal.PrintMatrix();
  eigenVec.PrintMatrix();
End
```

End

End is the final command each compound script must have (unless there is an *EndRun* command (see *EndRun*)). After *Compound* executes what is written in the script then it passes control again to normal ORCA input reading. ORCA will continue analyze the input that rests after the *Compound* part but it will not run any calculation.

EndRun

#EndRun* is an alternative to the *End* command (see *EndRun*) for ending the execution of a *Compound* script. The difference between *end* and *EndRun* is that *EndRun* ignores everything after the *Compound* block. This makes it even easier to use *Compound* as a full workflow run.

EndFor

All *For* loops (see *For*) must finish with *EndFor*. The syntax and an example is shown in the *For* section (see *For*).

NOTE For *EndFor* both *EndFor* and *EndFor;* are possible.

For

For loops are used to perform repetitive tasks. The syntax is the following:

Syntax:

For *variable* **From** *Start value* **To** *End value* **Do**

commands

EndFor or **EndFor;**

Variable should be a variable name not previously defined. *Start value* and *End value* should be integers defining the start and end value of the *variable*. *Start value* and *End value* can be numbers, predefined variables or functions of previously defined variables. The only requirement is that they should be **integers**. Keep in mind that the loop will be performed from the first value to the *End value*, including the *End value*.

Example:

```
# --------------------------------------
# This is a script to check 'for' loops
# --------------------------------------

# ---------------------------------
# Some necessary initial definitions
# ---------------------------------
Variable x = {0.0, 1.0, 2.0, 3.0, 4.0};
Variable f;
Variable loopStart;
Variable upLimit;

# -----------------------------------
# Case 1.
# Constant Start / Constant End
# -----------------------------------
print(" --------------  Case 1 --------------\n");
print("     Constant Start / Constant End    \n");
print("          f = index*x[index]          \n");
print( "      for index from 0 to 4 Do     \n");
print(" -------------------------------------\n");
for index from 0 to 4 Do
  f = index*x[index];
  print("Index: %3d    x[index]:  %.2lf   f:   %.2lf\n", index, x[index], f);
EndFor

loopStart = 0;
upLimit   = 4;
print(" --------------  Case 2 --------------\n");
print("     Variable Start / Variable End    \n");
print("          f = index*x[index]          \n");
```

(continues on next page)

(continued from previous page)

```
print( "   for index from loopStart to upLimit Do\n");
print(" -----------------------------------\n");
for index from loopStart to upLimit Do
  f = index*x[index];
  print("Index: %3d    x[index]:  %.2lf   f:   %.2lf\n", index, x[index], f);
EndFor


loopStart  = 1;
upLimit    = 3;
print(" --------------  Case 3 --------------    \n");
print("     function Start / function End        \n");
print("          f = index*x[index]              \n");
print( "   for index from start-1 to upLimit+1 Do\n");
print(" -----------------------------------    \n");
for index from loopStart-1 to upLimit+1 Do
  f = index*x[index];
  print("Index: %3d    x[index]:  %.2lf   f:   %.2lf\n", index, x[index], f);
EndFor

End
```

Geometry

In *Compound* we have *Geometry* objects. These objects can be treated like normal variables of type '*compGeometry*'. An important difference between normal variables and *Geometry* variables is the declaration. Instead of the normal:

```
Variable myGeom;
```

we excplicitly have to declare that this is a geometry. So the syntax for a geometry declaration is:

Syntax:

```
Geometry myGeom;

Geometry myGeom1, myGeom2;
```

Using the second definition one can define two geometry objects in the same line.

Below is a list of functions that work on *Geometry* objects.

- BohrToAngs(*G.BohrToAngs*)
- CreateBSSE (*G.CreateBSSE*)
- CreateFragments(*G.CreateFragments*)
- DisplaceAtom(*G.DisplaceAtom*)
- FollowNormalMode (*G.FollowNormalMode*)
- GetAngle(*G.GetAngle*)
- GetAtomicNumbers(*G.GetAtomicNumbers*)
- GetBondDistance(*G.GetBondDistance*)
- GetCartesians(*G.GetCartesians*)
- GetGhostAtoms(*G.GetGhostAtoms*)
- GetNumOfAtoms(*G.GetNumOfAtoms*)
- MoveAtomToCenter(*G.MoveAtomToCenter*)
- Read(*G.Read*)

- RemoveAtoms(*G.RemoveAtoms*)
- RemoveElements(*G.RemoveElements*)
- WriteXYZFile(*G.WriteXYZFile*)

G.BohrToAngs

BohrToAngs command acts on a geometry object (see see *Geometry*). It will transform the geometry of the loaded geometry object from Bohr to Angstroms. Practically it will just multiply the coordinates with the factor *0.529177249*.

Syntax:
myGeom.BohrToAngs();

Where:

- *myGeom* is a geometry object that already contains a geometry

Example:

```
# ----------------------------------------------------
# This is a script to check the BohrToAngs function
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable CC;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  myGeom.BohrToAngs();
  CC = myGeom.GetCartesians();
  CC.PrintMatrix();
End
```

G.CreateBSSE

CreateBSSE command acts on a geometry object (see see *Geometry*). In the case that the geometry object contains *ghost* atoms then *CreateBSSE* will create five new files:

- myFilename_FragmentA.xyz
- myFilename_MonomerA.xyz
- myFilename_FragmentB.xyz
- myFilename_MonomerB.xyz
- myFilename_Total.xyz

Syntax:
myGeom.CreateBSSE(filename=myFilename);

Where:

- *myGeom* is a geometry object that already contains a geometry
- *filename* is a base filename for the created files.

Example:

```
# -------------------------------------------------------
# This is a scrtipt to check the geom.CreateBSSE command
# -------------------------------------------------------

*xyz 0 1
o:      -1.69296787   -0.05579265    0.00556629
h:      -2.01296504    0.84704339   -0.01586469
h:      -0.73325076    0.04238910    0.00084302
o        1.23009925    0.02698440   -0.00375550
h        1.60672086   -0.41139567    0.76236888
h        1.60236356   -0.44922858   -0.74915800
*

%Compound
  Geometry monomerA;
  variable myFilename     = "BSSE";
  Variable method         = "BP86";

  # -------------------------------------
  # Calculation for Fragment A
  # -------------------------------------
  New_Step
    !&{method}
  Step_End

  # -------------------------------------
  # Read the geometry of Fragment A
  # -------------------------------------
  monomerA.Read();

  # -------------------------------------
  # Create the missing xyz files
  # -------------------------------------
  monomerA.CreateBSSE(filename=myFilename);

End
```

NOTE The files will contain XYZ geometries in **BOHRS**.

G.CreateFragments

CreateBSSE command acts on a geometry object (see see *Geometry*). In the case that the geometry object contains *ghost* atoms then *CreateBSSE* will create five new files:

- myFilename_FragmentA.xyz
- myFilename_MonomerA.xyz
- myFilename_FragmentB.xyz
- myFilename_MonomerB.xyz
- myFilename_Total.xyz

Syntax:
myGeom.CreateBSSE(filename=myFilename);

Where:

- *myGeom* is a geometry object that already contains a geometry
- *filename* is a base filename for the created files.

Example:

```
# -------------------------------------------------------
# This is a scrtipt to check the geom.CreateBSSE command
# -------------------------------------------------------

*xyz 0 1
o:      -1.69296787   -0.05579265    0.00556629
h:      -2.01296504    0.84704339   -0.01586469
h:      -0.73325076    0.04238910    0.00084302
o        1.23009925    0.02698440   -0.00375550
h        1.60672086   -0.41139567    0.76236888
h        1.60236356   -0.44922858   -0.74915800
*

%Compound
  Geometry monomerA;
  variable myFilename     = "BSSE";
  Variable method         = "BP86";

  # -------------------------------------
  # Calculation for Fragment A
  # -------------------------------------
  New_Step
    !&{method}
  Step_End

  # -------------------------------------
  # Read the geometry of Fragment A
  # -------------------------------------
  monomerA.Read();

  # -------------------------------------
  # Create the missing xyz files
  # -------------------------------------
  monomerA.CreateBSSE(filename=myFilename);

End
```

NOTE The files will contain XYZ geometries in **BOHRS**.

G.FollowNormalMode

FollowNormalMode command acts on a geometry object (see see *Geometry*). It will displace the loaded geometry following a chosen normal mode of vibtation

Syntax:
myGeom.FollowNormalMode(vibrationSN=myVibration, [ScalingFactor=myScalingFactor]);

Where:

- *myGeom* is a geometry object that already contains a geometry
- *vibrationSN* is the serial number of the vibration. **NOTE** Please remember that counting starts with 1.
- *scalingFactor* is the scaling of the normal mode of vibration. This argument is optional.

Example:

```
# -----------------------------------------------------
# This is a script to check the followNormalMode function
# -----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
```

(continues on next page)

(continued from previous page)

```
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable CC, normalModes;
  Variable res = -1;
  New_Step
    !BP86 Freq
  Step_End
  myGeom.Read();
  myGeom.FollowNormalMode(vibrationSN=7, scalingFactor=0.8);
  CC = myGeom.GetCartesians();
  CC.PrintMatrix();
End
```

G.DisplaceAtom

Function *DisplaceAtom* acts on geometry objects and moves the cartesian coordintates of a given atom of the current geometry. The displacement is in Bohrs.

Syntax:

res = geom.DisplaceAtom(atomID, dx, dy, dz)

Where:

geom A geometry object previously loaded.

atomID The index of the atom we want to move. Please note that counting starts with zero.

dx The displacement in the x axis in Bohrs.

dy The displacement in the y axis in Bohrs.

dz The displacement in the z axis in Bohrs.

Example:

```
# -----------------------------------
# This is to test Geometry function
#         DisplaceAtom
# -----------------------------------
%Compound
  Variable dist=0.1;
  Geometry myGeom;
  NewStep
    !BP86
    *xyz 0 1
       O  0.000  0.000   0.394
       H -0.755  0.000  -0.197
       H  0.755  0.000  -0.197
    *
  StepEnd
  myGeom.Read();
  myGeom.DisplaceAtom(0, 0.0, 0.0, -dist);  #Move the z coordinate of Oxygen
  myGeom.WriteXYZFile(filename="displaced.xyz");

End
```

G.GetAngle

Function *GetAngle* acts on geometry objects and returns the angle between three atoms in **Degrees**.

Syntax:

res = geom.GetAngle(atomA, atomB, atomC)

Where:

res The angle between atoms atomA, atomB and atomC.

geom A geometry object previously loaded.

atomA The index of atomA in the geometry.

atomB The index of atomB in the geometry.

atomC The index of atomC in the geometry.

NOTE Please not that the ordering of atoms is important

Example:

```
# ------------------------------------
# This is to test Geometry function
#           GetAngle
# ------------------------------------
%Compound
  Variable dist;
  Geometry myGeom;
  New_Step
    !BP86
    *xyz 0 1
       O  0.00000000  0.00000000   0.39393904
       H -0.75503878  0.00000000  -0.19696952
       H  0.75503878  0.00000000  -0.19696952
    *
  Step_End
  myGeom.Read();  #Reads teh geometry of the previous step
  print( " -------------------------------------------------------\n");
  print( "  Compound Geometry functions test (GetAngle) \n");
  print( "  It should print 103.9051 Degrees\n");
  print( " -------------------------------------------------------\n");
  print( " The angle between atom %d, atom %d and atom %d  is: %.4lf Degreess\n",
           1, 0, 2, myGeom.GetAngle(1,0,2));

End
```

G.GetAtomicNumbers

Function *GetAtomicNumbers* acts on geometry objects and returns and array with the atomic numbers of the elements in the working geometry.

Syntax: *atomNumbers = geom.GetAtomicNumbers()*

atomNumbers A variable that will be filled with the values of the atomic numbers

geom A geometry object that should already be loaded.

Example:

```
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
```

(continues on next page)

(continued from previous page)

```
*

%Compound
  Geometry myGeom;
  Variable atomicNumbers;

  New_Step
    !BP86
  Step_End
  myGeom.Read();
  atomicNumbers = myGeom.GetAtomicNumbers();
  print("\nCompound \n");
  for i from 0 to atomicNumbers.GetSize()-1 Do
    print("Atom '%d': atomic number: %d\n", i, atomicNumbers[i]);
  EndFor
End
```

G.GetBondDistance

Function *GetBondDistance* acts on geometry objects and returns the distance between two atoms **in Bohrs**.

Syntax:

res = geom.GetBondDistance(atomA, atomB)

Where:

res The distance between atoms atomA and atomB.

geom A geometry object previously loaded.

atomA The index of atomA in the geometry.

atomB The index of atomB in the geometry.

NOTE indices start counting from 0

Example:

```
# ------------------------------------
# This is to test Geometry function
#    GetBondDistance
# ------------------------------------
%Compound
  Variable dist;
  Geometry myGeom;
  New_Step
    !BP86
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
  myGeom.Read();  #Reads teh geometry of the previous step
  print( " -------------------------------------------------------\n");
  print( "  Compound Geometry functions test (GetBondDistance) \n");
  print( "  It should print 1.5118\n");
  print( " -------------------------------------------------------\n");
  print( " The distance between atom %d and atom %d  is: %.4lf Bohr\n",
           0, 1, myGeom.GetBondDistance(0,1));

End
```

G.GetCartesians

Function *GetCartesians* acts on geometry objects (see *Geometry*) and returns the distance xyz cartesian coordinates. Please remember that it alsways returns the cooridnates in **BOHRS**.

Syntax:

coords = geom.GetCartesians()

Where:

coords: A (*nAtoms*,3) array with the cartesian coordinates in **BOHRS**.

geom: A geometry object previously loaded.

Example:

```
# ----------------------------------------------------
# This is a script to check the GetCartesians function
# NOTE: It always return it in Bohrs!
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable CC;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  CC = myGeom.GetCartesians();
  for i from 0 to CC.GetDim1()-1 Do
    print("%12.9lf  %12.9lf  %12.9lf\n",
    CC[i][0], CC[i][1], CC[i][2]);
  endFor
End
```

G.GetGhostAtoms

Function *GetGhostAtoms* acts on geometry objects (see *Geometry*). It returns a vector of size nAtoms where for each atom the value will be -1 if it is a ghost atom, otherwise the atomic number of the element

Syntax:

ghostAtoms = geom.GetGhostAtoms()

Where:

ghostAtoms: A (*nAtoms*,1) integer vector with values -1 or the atomic number of the atom, in case it is not a ghost atom.

geom: A geometry object previously loaded.

Example:

```
# ----------------------------------------------------
# This is a script to check the getGhostAtoms function
# ----------------------------------------------------
*xyz 0 1
o:      -1.69296787   -0.05579265    0.00556629
h:      -2.01296504    0.84704339   -0.01586469
```

(continues on next page)

(continued from previous page)

```
h:      -0.73325076    0.04238910    0.00084302
o        1.23009925    0.02698440   -0.00375550
h        1.60672086   -0.41139567    0.76236888
h        1.60236356   -0.44922858   -0.74915800
*

%Compound
  Geometry myGeom;
  Variable ghostAtoms;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  ghostAtoms = myGeom.GetGhostAtoms();
  ghostAtoms.PrintMatrix();
End
```

G.GetNumOfAtoms

GetNumOfAtoms returns an integer with the number of atoms of the working geometry.

Syntax:

res = geom.GetNumOfAtoms();

Where:

res is the resulting number of atoms

geom is the name of a geometry variable (see *Geometry*) we are using.

Example:

```
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable numOfAtoms = 0;

  New_Step
    !BP86
  Step_End
  Alias currStep;
  myGeom.Read(currStep);
  numOfAtoms = myGeom.GetNumOfAtoms();
  print("\nCompound \n");
  print("Number of atoms: %d (it should print 3)\n", numOfAtoms);

End
```

G.MoveAtomToCenter

Function *MoveAtomToCenter* acts on geometry objects (see *Geometry*). It will adjust the cartesian coordinates so that the chosen atom will rest at (0.0 0.0 0.0).

Syntax:

geom.MoveAtomToCenter(atom serial nubmer);

Where:

geom: A geometry object previously loaded.

atom serial number: The serial number of the atom in the geometry.

Example:

```
# ----------------------------------------------------
# This is a script to check the moveAtomToCenter function
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable CC;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  myGeom.MoveAtomToCenter(0);
  myGeom.BohrToAngs();
  CC = myGeom.GetCartesians();
  CC.PrintMatrix();
End
```

NOTE Please remember that counting **starts with 0**, meaning that the first atom is 0 and not 1!

G.Read

Function *Read* acts on geometry objects (see *Geometry*) and reads a geometry from a property file related to a previous step.

Syntax:

geom.Read([stepID=myStepID], propertySN=myPropertySN)

Where:

geom: A geometry object that will be updated.

stepID: The step from which we are going to read the geometry. If not given the previous step will be used.

propertySN: The serial number the geometry in the property file. If not given the last available geometry will be used.

Example:

```
# ----------------------------------------------------
# This is a script to check the Read function for
#   geometries
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
```

(continues on next page)

(continued from previous page)

```
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable CC;
  New_Step
    !BP86 opt
  Step_End
  myGeom.Read(propertySN=2);
End
```

G.RemoveAtoms

Function *RemoveAtoms* acts on geometry objects (see *Geometry*). It accepts a list of atoms and removes them from the loaded geometry. In the end the geometry object will be updated.

Syntax:

geom.RemoveAtoms(atom1, atom2, …);

Where:

geom: A geometry object previously loaded.

atom1, atom2, …: The serial number of the atoms in the geometry.

Example:

```
# ----------------------------------------------------
# This is a script to check the RemoveAtoms function
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable numOfAtoms;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  numOfAtoms = myGeom.GetNumOfAtoms();
  print("Number of atoms before: %d (It should print 3)\n", numOfAtoms);
  myGeom.RemoveAtoms(0);  #Remove the first atom
  numOfAtoms = myGeom.GetNumOfAtoms();
  print("Number of atoms after : %d (It should print 2)\n", numOfAtoms);
End
```

NOTE Please remember that counting **starts with 0**, meaning that the first atom is 0 and not 1!

G.RemoveElements

Function *RemoveElements* acts on geometry objects (see *Geometry*). It will remove from the loaded geometry all atoms with an atomic number given in the list.

Syntax:

geom.RemoveElements(atomNumber1, atomicNumber2, …);

Where:

geom: A geometry object previously loaded.

atomicNumber1, atomicNumber2, …: The atomic number of elements to be removed from the current geometry.

Example:

```
# ----------------------------------------------------
# This is a script to check the RemoveElements function
# ----------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable numOfAtoms;
  Variable CC;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  numOfAtoms = myGeom.GetNumOfAtoms();
  CC = myGeom.GetCartesians();
  CC.PrintMatrix();
  print("Number of atoms before: %d (It should print 3)\n", numOfAtoms);
  myGeom.RemoveElements(8);  #Remove the oxygen
  numOfAtoms = myGeom.GetNumOfAtoms();
  print("Number of atoms after : %d (It should print 2)\n", numOfAtoms);
  CC = myGeom.GetCartesians();
  CC.PrintMatrix();
End
```

G.WriteXYZFile

Function *WriteXYZFile* acts on geometry objects (see *Geometry*) and writes on disc an xyz file with the coordinates of the current goemetry object. Please remember that by default it writes the coordinates in **Angstroems**.

Syntax:

res = geom.WriteXYZFile(filename=myFilename, [useBohr=True/False])

Where:

res: An integer that returns '0' if everything worked smoothly.

myFilename The name of the file that will contain the coordinates.

useBohr: A boolean that can take the value of "True" or "False". This is an optional argument and the default value is "False".

geom: A geometry object previously loaded.

NOTE: Please note that we changed the default units. Now by default we write the xyz file in Angstroems. We added the parameter UseBohr that is set by default to "False" but if it is used with the value "True" then the geometry is saved in Bohrs.

Example:

```
# -------------------------------------------------------
# This is a script to check the WriteXYZ function
# NOTE: By default it write the coordinates in Angstroems!
# -------------------------------------------------------
*xyz 0 1
  O      -1.69296787   -0.05579265    0.00556629
  H      -2.01296504    0.84704339   -0.01586469
  H      -0.73325076    0.04238910    0.00084302
*

%Compound
  Geometry myGeom;
  Variable res=-1;
  New_Step
    !BP86
  Step_End
  myGeom.Read();
  res = myGeom.WriteXYZFile(filename="myGeom.xyz",useBohr=false);
End
```

GOAT

In *Compound* we have *GOAT* objects. These objects can be treated like normal variables of type '*compGOAT*'. An important difference between normal variables and *GOAT* variables is the declaration. Instead of the normal:

```
Variable myGoat;
```

we excplicitly have to declare that this is a GOAT object. So the syntax for a goat declaration is:

Syntax:

```
GOAT myGoat;
```

Below is a list of functions that work on *Geometry* objects.

- Get_Energy (*Goat.Get_Energy*)
- Get_Num_Of_Geometries (*Goat.Get_Num_Of_Geometries*)
- ParseEensembleFile (*Goat.ParseEnsembleFile*)
- Set_Basename (*Goat.Set_Basename*)
- Print (*Goat.Print*)
- WriteXYZFile (*Goat.WriteXYZFile*)

Example:

```
# Example for the usage of GOAT objects in compound
#
# NOTE:  The file 'test.finalensemble.xyz'
#        should be availablie in the directory
%Compound
  goat myGoat;
  variable myFilename;
  myGoat.Set_Basename("test");
  myGoat.ParseEnsemblefile();
```

(continues on next page)

(continued from previous page)

```
  for myGeomID from 1 to myGoat.Get_Num_of_Geometries() Do
    write2String(myFilename, "goatGeom_%d.xyz", myGeomID);
    myGoat.WriteXYZFile(geomID=myGeomID, filename=myFilename);
    print("geomID: %d Energy: %lf\n", myGeomID, myGoat.Get_
↪Energy(geomID=myGeomID));
  EndFor
  myGoat.Print();
EndRun
```

Goat.Get_Energy

Function *Get_Energy* acts on goat objects (see *GOAT*) and returns a double with the energy of the requested structure.

Syntax:

res = myGoat.Get_Energy(geomID=myGeomID)

Where:

res: A double with the energy of the requested structure.

myGoat A GOAT object that is already loaded (meaning it has already parse an ensemble file, see (see *Goat.ParseEnsembleFile*).

myGeomID: The geometry ID of the requested structure. Counting starts from 1.

Example:

see example in *GOAT*

Goat.Get_Num_Of_Geometries

Function *Get_Num_Of_Geometries* acts on goat objects (see *GOAT*) and returns an integer with the number of structures available in the current *GOAT* object.

Syntax:

res = myGoat.Get_Num_Of_Geometries()

Where:

res: An integer with the number of geometries in the current *GOAT* object.

myGoat A GOAT object that is already loaded (meaning it has already parse an ensemble file, see (see *Goat.ParseEnsembleFile*).

Example:

see example in *GOAT*

Goat.ParseEnsembleFile

Function *ParseEnsembleFile* acts on goat objects (see *GOAT*). It will read an ensemble file created by a GOAT run and then update the object with the geometries and energies of each geometry available in the current *GOAT* object.

Syntax:

res = geom.ParseEnesembleFile(filename=myFilename)

Where:

res: An integer that returns '0' if everything worked smoothly.

myFilename The name of the file that will contain the coordinates.

myGoat: A GOAT object previously loaded.

Example:

see example in *GOAT*

NOTE: The old syntax (Parse_Ensemble_File) is still compatible but please do not use it because in the future it will be deprecated.

Goat.Set_Basename

Function *Set_Basename* acts on goat objects (see *GOAT*). It set the basename of the ensemble file created by a GOAT run. It will automatically add the extension: `'.finalensemble.xyz'` in the end of the basename.

Syntax:

myGoat.Set_Basename(myFilename)

Where:

myFilename The basename of the GOAT ensemble file (**without the extension 'finalensemble.xyz'**).

myGOAT: A GOAT object previously loaded.

Example:

see example in *GOAT*

Goat.Print

Function *Print* acts on goat objects (see *GOAT*). It will print in the output the geometries of current GOAT object.

Syntax:

myGoat.Print()

Where:

myGoat: A GOAT object previously loaded.

Example:

see example in *GOAT*

Goat.WriteXYZFile

Function *WriteXYZFile* acts on goat objects (see *GOAT*). It will write the requested geometry to an XYZ file on disk. It expects two arguments:

1. The geometry ID of the geometry and
2. The filename that it will use to store the geometry

Syntax:

myGoat.WriteXYZFile(geomID=myGeomID, filename=myFilename)

Where:

myFilename: The name that it will be used to store the geometry on the disk.

myGeomID: The geometry ID of the requested structure. Counting starts from 1.

Example:

see example in *GOAT*

GetNumOfInstances

The *GetNumOfInstances* returns the number of instances of a specific object in a propertyfile.

Syntax:

[res=] GetNumOfInstances(propertyName=myName, [step=myStep], [filename=myFilename], [baseProperty=true/false])

Where:

res: An integer that returns the number of instances of the required property in the property file.

propertyName: A string alias that defines the variable the user wants to read.

step: The step from which we want to read the property. If not given the property file from the last step will be read.

filename: A filename of a property file. If a filename and at the same time a step are provided the program will ignore the step and try to read the property file with the given filename.

NOTE please note that in the end of filename the extension *.property.txt* will be added.

baseProperty: A true/false boolean. The default value is set to false. If the value is set to true then a generic property of the type asked will be read. This means if dipole moment is asked, it will return the last dipole moment, irrelevant if and MP2 or SCF one wad defined.

Example

```
# ----------------------------------------------------
# This is an example script for readNumOfInstances
# ----------------------------------------------------
%Compound
  Variable res = 0;
  Variable myProperty="MP2_DIPOLE_TOTAL";
  Variable myBaseProperty="DIPOLE_MOMENT_TOTAL";
  New_Step
    !MP2
    #%mp2
    #  density relaxed
    #end
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
  # First read the MP2 dipole moment
  res = GetNumOfInstances(propertyName=myProperty);
  print("Num of MP2 dipole moments   : %d\n", res);
  res = GetNumOfInstances(propertyName=myBaseProperty, Property_Base=true);
  print("Num of total dipole moments : %d\n", res);
End
```

GoTo

The *GoTo* command allows the 'jump' inside the normal flow of a *compound* script. The syntax of the command can be best presented through an example.

Example:

```
# ----------------------------------------------------
# This is an example script for GoTo
# (It should print only 0,1,2,3)
# ----------------------------------------------------
%Compound
```

(continues on next page)

(continued from previous page)

```
  Variable TCut=3;
  Variable Done;
  for i from 0 to 6 Do
    print("Index: %d\n", i);
    if (i >= TCut) then
      GoTo Done;
    EndIf
  EndFor
  Done:
    print("Done\n");
End
```

Please note that the variable we use as a label for the *GoTo* command should be previously defined like a normal variable.

If

The *if* block allows the user to make decisions. The syntax in *Compound* is the following:
Syntax:

If *(expression)* **Then**

actions

Else if *(expression)* **Then**

actions

Else

actions

Endif

Below is an example of the usage of *if block* in *compound*.

```
# ------------------------------------------------------------
# This is to check all available ways of 'if blocks'
# ------------------------------------------------------------
Variable x1 = 10.0;
Variable y1 = 20.0;
Variable b1 = False;
Variable b2 = True;
Variable s1 = "alpha";
Variable s2 = "beta";
Variable s3 = "alpha";
print( " ------------------------------------------------------------ \n");
print( " ----------      SUMMARY OF IF CASES        -------------- \n");
print( " ------------------------------------------------------------ \n");

print(" x1: %.1lf\n", x1);
print(" y1: %.1lf\n", y1);
print(" b1: %s\n", b1.GetString());
print(" b2: %s\n", b2.GetString());
print(" s1: %s\n", s1);
print(" s2: %s\n", s2);
print(" s3: %s\n", s3);
# ****************************************************************
#                           DOUBLES
# ****************************************************************
print(" -------------------     Doubles     ---------------------- \n");
print("      Variable/constant / One operator  / if (x1>5)         \n");
print("                    No else if/No else                      \n");
```

(continues on next page)

(continued from previous page)

```
if (x1>5) then
  print(" %.2lf > 5 \n", x1);
endif
# ----------------------------------------------------------------
print("      function / Variable / One operator  / if (3*x1>y1)    \n");
print("                   no else if /   else                      \n");
if (3*x1>y1) then
  print(" 3*%.1lf > %.1lf\n", x1, y1);
else
  print(" 3*%.1lf < %.1lf\n", x1, y1);
endif
# ----------------------------------------------------------------
print("     function / function / One operator  / if (x1-y1>-10.0) \n");
print("                     else if/else                      \n");
if (x1-y1>-10.0) then
  print(" %.2lf - %.2lf > -10.0\n", x1, y1);
else if (x1-y1 < -10.0) then
  print(" %.2lf - %.2lf < -10.0\n", x1, y1);
else
  print(" %.2lf - %.2lf = -10.0\n", x1, y1);
endif
# ****************************************************************
#                          BOOLEANS
# ****************************************************************
print(" ---------------      Booleans      ---------------------------\n");
print("       Variable  /  No operator  /  if (b1)                   \n");
if (b1) then
  print("b1 is True\n");
else
  print("b1 is False\n");
endIf
# ----------------------------------------------------------------------
# ----------------------------------------------------------------------
print("       Constant  / No operator  / if (true)                   \n");
if (True) then
  print( "True\n");
else
  print( "False");
endIf
# ----------------------------------------------------------------------
# ----------------------------------------------------------------------
print("       Variable/Variable / AND operator / if (b1 and b2)                  \n
↪");
if (b1 and b2) then
  print("(%s and %s) is true\n", b1.GetString(), b2.GetString());
else
  print("(%s and %s) is not true\n", b1.GetString(), b2.GetString());
endIf
# ----------------------------------------------------------------------
# ----------------------------------------------------------------------
print("       Variable/Variable / OR operator / if (b1 or b2)                 \n");
if (b1 OR b2) then
  print("(%s or %s) is true\n", b1.GetString(), b2.GetString());
else
  print("(%s or %s) is not true\n", b1.GetString(), b2.GetString());
endIf
# ----------------------------------------------------------------------
# ----------------------------------------------------------------------
print("       Bool /Doubles Function / AND operator / if (b1 and x1>y1 )\n");
if (b1 and y1>x1) then
  print("(%s and %.1lf>%.1lf) is True\n", b1.GetString(), x1, y1);
```

(continues on next page)

(continued from previous page)

```
else
  print("(%s and %.1lf>%.1lf) is False\n", b1.GetString(), x1, y1);
endIf
# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
print("       Nested if / if (b2) then if (y1>x1)\n");
if (b2) then
  if (y1 > x1) then
    print ( "(%s is True) and (%.1lf>%.1lf)\n", b2.GetString(), y1, x1);
  else
    print ( "(%s is True) and (%.1lf<%.1lf)\n", b2.GetString(), y1, x1);
  endIf
else
    print ( "%s is False\n", b2.GetString());
endIf
# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
print(" ----------------      Strings     --------------------------\n");
print(" -------------------------------------------------------------\n");
print(" -------------------------------------------------------------\n");
print("       Variable/Variable / if s1=s2\n");
if (s1=s2) then
  print("%s is same as %s \n", s1, s2);
else
  print("%s is not same as %s \n", s1, s2);
EndIf

# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
print(" ----------------      Strings     --------------------------\n");
print(" -------------------------------------------------------------\n");
print(" -------------------------------------------------------------\n");
print("       Variable/constant / if s1=\"alpha\"\n");
if (s1="alpha") then
  print("%s is same as %s \n", s1, "alpha");
else
  print("%s is not same as %s \n", s1, "alpha");
EndIf
End
```

Some comments about the syntax:

The *Else if* or *Else* blocks are not obligatory.

The numerical operators that can be used are: '>', '<', '>=', '<=', '='.

The available logical operators are: *'and' and 'or'*.

Unfortunately in the current version multi-parentheses are not allowed.

There is now the possibility to compare strings.

InvertMatrix

Compound can peform matrix algebraic operations, one of the available algebraic operation is the inversion of a matrix. Be carefull that the matrix, whose the invert we are looking for, **must be** a real, square matrix.

Syntax:
AInvert = A.InvertMatrix();

Where:

- *A:* The matrix to be inverted.
- *AInvert:* The invert of A. It can be A itself and then A will just be updated.

Example:

```
# ----------------------------------------------------
# This is an example script for matrix inversion
# ----------------------------------------------------
%Compound
  Variable Dim=3;
  Variable A[Dim][Dim];
  Variable invertA, C;
  Variable res=-1;
  for i from 0 to Dim-1 Do
    for j from 0 to Dim-1 Do
      if (i=j) then
        A[i][j] = i+1;
      else
        A[i][j] =  0.0;
      EndIf
    EndFor
  EndFor
  invertA = A.invertMatrix();
  A.PrintMatrix();
  invertA.PrintMatrix();
  C = Mat_x_Mat(A,invertA,false, false, 1.0, 1.0);
  C.PrintMatrix();
End
```

Mat_p_Mat

Compound can peform matrix algebraic operations, one of the available algebraic operation is matrix addittion. In order to add two matrices they **must** have the same dimensions.

Syntax:
C=Mat_p_Mat(alpha, A, beta, B);

Where:

- *C:* The resulting matrix.
- *alpha:* The coefficient for matrix A.
- *A:* The left matrix of the addition.
- *beta:* The coefficient for matrix B.
- *B:* The right matrix of the addition.

Example:

```
# ----------------------------------------------------
# This is an example script for matrix addition
# ----------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
%Compound
  Variable Dim=3;
  Variable A[Dim][Dim];
  Variable B[Dim][Dim];
  Variable C;
  Variable res=-1;
  for i from 0 to Dim-1 Do
    for j from 0 to Dim-1 Do
      A[i][j] = 1.0;
      B[i][j] = 2.0;
    EndFor
  EndFor
  A.PrintMatrix();
  B.PrintMatrix();
  C = Mat_p_Mat(2.0, A, 3.0, B);
  C.PrintMatrix();
End
```

Mat_x_Mat

Compound can peform matrix algebraic operations, one of the available algebraic operation is matrix multiplication. In general we can multiply each matrix with constants *alpha* and *beta* so that the general multiplication is:

*C=(alpha*A)*(beta*B)

In addition each of matrices A and B are allowed to be transposed.

Syntax:
C=Mat_x_Mat(A, B, [transposeA], [transposeB], [alpha], [beta]);

Where:

- *C:* The resulting matrix.
- *A:* The left matrix of the multiplication.
- *B:* The right matrix of the multiplication.
- *transposeA:* A boolean to state if matrix A should be transposed before the mutliplication (default: False).
- *transposeB:* A boolean to state if matrix B should be transposed before the multiplication (default: False).
- *alpha:* A scalar to multiply matrix A before the mutliplication (default 1.0).
- *beta:* A scalar to mutliply matrix B before the multiplication (default 1.0).

Example:

```
# ----------------------------------------------------
# This is an example script for matrix multiplication
# ----------------------------------------------------
%Compound
  Variable Dim=3;
  Variable A[Dim][Dim];
  Variable invertA;
  Variable C;
  Variable D;
  Variable res=-1;
  for i from 0 to Dim-1 Do
    for j from 0 to Dim-1 Do
      if (i=j) then
        A[i][j] = i+1;
      else
        A[i][j] =  0.0;
```

(continues on next page)

(continued from previous page)

```
      EndIf
    EndFor
  EndFor
  A.invertMatrix(invertA);
  A.PrintMatrix();
  invertA.PrintMatrix();
  C = Mat_x_Mat(A,invertA,false, false, 1.0, 1.0);
  C.PrintMatrix();
  C = Mat_x_Mat(A, invertA, true, true, 1.0, 1.0);
  C.PrintMatrix();
  C = Mat_x_Mat(A, invertA, false, false, 2.0);
  C.PrintMatrix();
  C = Mat_x_Mat(A, invertA, false, false, 2.0, 3.0);
  C.PrintMatrix();
End
```

Mat_x_Scal

Compound can peform matrix algebraic operations, one of the available algebraic operation is multiplication of the elements of a matrix with a scalar. The function returns the multiplied matrix that can be the one that we use as an argument in the parenthesis, meaning it is updated, or a different one.

Syntax:
C=Mat_x_Scal(alpha, A);

Where:

- *C:* The resulting matrix.
- *alpha:* A scalar to multiply the elements of matrix A.
- *A:* The matrix to be mutliplied.

Example:

```
# ----------------------------------------------------
# This is an example script for matrix times scalar
# ----------------------------------------------------
%Compound
  Variable Dim=3;
  Variable A[Dim][Dim];
  Variable alpha=2.0;
  Variable C;
  for i from 0 to Dim-1 Do
    for j from 0 to Dim-1 Do
        A[i][j] = i+j;
    EndFor
  EndFor
  A.PrintMatrix();
  C = Mat_x_Scal(alpha,A);
  A = Mat_x_Scal(alpha,A);
  A.PrintMatrix();
  C.PrintMatrix();
End
```

NewGeom

NewGeom is a platform for geometry manipulation. The basic idea is to have functions that can read a geometry and then produce one or more new geometries with some characteristics that we need. For the moment under the umbrella of *NewGeom* fall 4 different functions, and these are: Displace, Remove_Atom, Remove_Element.

NOTE: The old syntax (New_Geom) is still compatible but please do not use it because in the future it will be deprecated.

Displace

The idea behind "*Displace*" is to have a structure, perform an analytical frequncy calculation on it (currently we do not store numerical frequencies in the property file) and then read the Hessian from this calculation to adjust the geometry based on a normal mode of vibration that we choose. The syntax of this command is:

syntax: NewGeom = (Displace, step, hessian, frequency, scaling)

where:
step is the step from which we choose the original geometry
hessian is the hessian read from a property file
frequency defines which normal mode we will use and
scaling is a factor of how severe we want the displacement to be.

We should note that *Displace* is the only command of the *new_geom* family of commands that will not store a geometry on disk but only internally pass the new geometry to the next calculation. An example of the usage of this command can be found in the script 'iterativeOptimization' that is given with ORCA. The relevant part is as follows: **example:**

```
# Define variables
                Variable MaxNTries    =  25;
                Variable CutOff       = -50;
                Variable displacement = 0.6;
                Variable NNegative =   0;
                Variable freqs[];
                Variable modes[];
                Variable NFreq;
                Variable limit;
                Variable done;
                Variable FinalEnergy;

                # ============================================================
                # Start a for loop over number of tries
                # ============================================================
                For itry From 1 To maxNTries Do

                # ----------------------------------
                # Run a geometry optimization
                # ----------------------------------
                New_Step
                ! tightopt freq verytightscf nopop def2-TZVP xyzfile
                Step_End
                Read freqs  = THERMO_FREQS[itry];
                Read modes  = HESSIAN_MODES[itry];
                Read NFreq  = THERMO_NUM_OF_FREQS[itry];
                limit = NFreq - 1;
                # ----------------------------------
                #  check for sufficeintly negative
                #  frequencies
                # ----------------------------------
                NNegative = 0;
                For ifreq From 0 to limit Do
                if ( freqs[ifreq] < CutOff )  then
                New_Geom = (Displace, itry,  modes,  ifreq, displacement);
```

OpenFile

Compound can write text files on disk. In order to write to a file a filepointer must be created. For this in *Compound* exists the command *OpenFile*.

Syntax:
filePtr = *OpenFile*(*Filename*, "*open mode*");

filePtr is a variable previously declared.

Filename can be a string or a variable of string type and represents the name of the file on disk.

There are two available *opening modes*:

- 'w'. In this mode a new file will be created and the user can write on it. If an old file with the same name exists, it's contents will be deleted.
- 'a'. In this mode if a file already exists, the user will append to what already exists in the file.

Example:

```
%Compound

# ------------------------------------------------------------
#      This is to check all available open and close
#                file options
# ------------------------------------------------------------
Variable myFilename    = "myFile.txt";
Variable file;

# --------------------------
# First open for writing
# --------------------------
file = openFile(myFilename, "w");
write2File(file, "This is the first time we write.\n");
closeFile(file);

# --------------------------
# Re-open to append
# --------------------------
file = openFile(myFilename, "a");
write2File(file, "This is the second time we write.\n");
closeFile(file);

End
```

Remove_Atom

Remove_atom removes an atom from a geometry given its index. We should point out that counting of atoms in ORCA starts with 0. After this command is executed it will store on a disk a new geomety in a xyz format where only the atom with the given index will be missing.

Syntax:

NewGeom = (Remove_atom, atomIndex, "filename", stepIndex, [geometry Index]);

where:
atom Index is the number of the atom we want to remove. It can be an integer number or a variable.
filename is the name of the file that we want to use for the new xyz file. It can be a string in quotation marks or a variable already defined before. In the name the xyz extension will be automatically appended. *step Index* the number of the step from which we will get the initial geometry. It has to be an integer number. *geometry Index* In case there are more than one geometries in the corresponding property file we can choose one. If no number is given but default the program will use the last one.

example:

if we use the normal ORCA input file:

```
*xyz 0 1
O         2.220871067         0.026716792         0.000620476
H         2.597492682        -0.411663274         0.766744858
H         2.593135384        -0.449496183        -0.744782026
*

%Compound "removeAtom.cmp"
```

together with the compound file "removeAtom.cmp" :

```
Variable filename  = "newGeom";
Variable atomIndex = 0;

New_Step
  !BP86
Step_End

New_Geom = ( Remove_atom, atomIndex, filename, 1);

end
```

then the xyz file 'newGeom.xyz' will be created that should look like:

```
2

   H         2.5974927       -0.4116633        0.7667449
   H         2.5931354       -0.4494962       -0.7447820
```

where the atom with *atomIndex = 0* meaning the first atom, meaning the oxygen is removed.

Remove_Element

Remove_element is similar to the Remove_atom but instead of using the index of the atom we use its atomic number. Thus the syntax is:

Syntax: NewGeom = (Remove_Element, atomic number, "filename", stepIndex, [geometry Index]);

where:
atomic number is the atomic number of the atom we want to remove. It can be an integer number or a variable. *filename* is the name of the file that we want to use for the new xyz file. It can be a string in quotation marks or a variable already defined before. In the name the xyz extension will be automatically appended. *step Index* the number of the step from which we will get the initial geometry. It has to be an integer number. *geometry Index* In case there are more than one geometries in the corresponding property file we can choose one. If no number is given but default the program will use the last one.

example:

if we use again the input from paragraph *1.1.7.2* but instead of asking the compound file "removeAtom.cmp" we ask for the compound file "removeElement.cmp" that looks like:

```
Variable filename  = "newGeom";

New_Step
  !BP86
Step_End

New_Geom = ( Remove_element, 8, filename, 1);

end
```

then we will get again the same xyz file that was crated in paragraph *Remove_Atom* since the atom with atomic number 8 (meaning the Oxygen) will be removed from the original geometry.

NewStep

NewStep signals the beginning of a new ORCA input.

Syntax:

NewStep
…Normal ORCA input commands
StepEnd

There is no restriction in the input of ORCA, except of course that it should not include another Compound block. It is important to remember that a *NewStep* command should always end with a *Step_End* command. Below we show a simple example.

NOTE: The old syntax (New_Step/Step_End) is still compatible but please do not use it because in the future it will be deprecated.

Example:

```
NewStep
! BP86 def2-SVP
StepEnd
```

There is only a basic fundamental difference with a normal ORCA input. Inside the *NewStep* block it is not necessary to include a geometry. ORCA will automatically try to read the geometry from the previous calculation. Of course a geometry can be given and then ORCA will use it.

Print

Printing in the ORCA outpug can be customized using the *print* command. The syntax of the *print* command closely follows the corresponding printf command from C/C++. So the usage of the *print* command is:

Syntax:
print(format string, [variables]);

For each variable there can be specifiers and flags for the specifiers. Currently *print* command supports three datatypes namely integers, doubles and strings.

A format specifier follows this prototype: *%[flags][width][.precision]specifier*

where details for the specifiers and flags can be found in table Table 8.1

Table 8.1: compound print Specifiers

Specifier	
s	strings
d	integers
lf	doubles
Flags	
number	width
.number	number of decimal digits
-	left alignement (by default is right)

Example:

```
# ------------------------------------------------------------
# This is to check all available print defintions
# ------------------------------------------------------------
Variable x1    = 2.0;
Variable x2    = {10.0, 20.0, 30.0, 40.0};
Variable x3    = {10,   20,   30,   40};
Variable x4    = {"ten", "twenty", "thirty", "fourty"};
Variable x5 = "test";
Variable index = 2;

print( " ---------------------------------------------- \n");
print( " ------ SUMMARY OF PRINT DEFINITIONS --------- \n");
print( " ---------------------------------------------- \n");
# ---------- No variables ------------
print( " No variables: \n" );
# ------------- Doubles --------------
print( " -------- Doubles --------------\n");
print( " constant double ( no format)              : %lf\n", 3.5);
print( " constant double (defined width)           : %16lf\n",3.5);
print( " constant double (defined width/accuracy) : %16.8lf\n", 3.5);
print( " variable double (x1)                      : %lf\n", x1);
print( " function double 2*x1*x1                   : %lf\n", 2*x1*x1);
print( " array element double                      : %lf\n", x2[2]);
print( " array element double with var index       : %lf\n", x2[index]);
#print( " array element double with function index : %lf\n", x2[index + 1];
# ------------ Integers --------------
print( " -------- Integers --------------\n");
print( " constant integer ( no format)             : %d\n", 3);
print( " constant integeer (defined width)         : %8d\n",3);
print( " variable integer (index)                  : %d\n", index);
print( " function integer 2*index*index            : %d\n", 2*index*index);
print( " array element                             : %d\n", x3[2]);
print( " array element intteger with var index     : %lf\n",x3[index]);
# ------------  Strings --------------
```

(continues on next page)

(continued from previous page)

```
print( " -------- Strings --------------\n");
print( " constant string ( no format)             : %s\n", "test");
print( " constant string (defined width)          : %8s\n","test");
print( " variable string                          : %s\n", x5);
print( " array element                            : %s\n", x4[2]);
print( " array element intteger with var index    : %s\n",x4[index]);
print(" ----------------------------------------------\n");
print(" ----------------------------------------------\n");
End
```

ReadProperty

One of the fundamental features of *Compound* is the ability to easily read ORCA calculated values from the property file.

Syntax:

[res=] readProperty(propertyName=myName, [stepID=myStep], [filename=myFilename], [baseProperty=true/false])

Where: res: An integer that returns the index of the found property if the property was found in the property file, -1 if the property does not exist. This is not obligatory.

propertyName: A string alias that defines the variable the user wants to read.

stepID: The step from which we want to read the property. If not given the property file from the last step will be read.

filename: A filename of a property file. If a filename and at the same time a step are provided the program will ignore the step and try to read the property file with the given filename.

NOTE please note that in the end of filename the extension *.property.txt* will be added.

baseProperty: A true/false boolean. The default value is set to false. If the value is set to true then a generic property of the type asked will be read. This means if dipole moment is asked, it will return the last dipole moment, irrelevant if and MP2 or SCF one wad defined.

Example

```
# ----------------------------------------------------
# This is an example script for readProperty
# ----------------------------------------------------
%Compound
  Variable enDirect=0.0;
  Variable enFilename=0.0;
  Variable myProperty="DFT_Total_en";
  Variable basename="compound_example_properertFile_readProperty";
  Variable newBasename ="newFilename";
  Variable res = -1;
  NewStep
    !BP86
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  StepEnd
  NewStep
    !B3LYP
  StepEnd
  # First read the energy directly
  res = enDirect.ReadProperty(propertyName=myProperty, stepID=2);
  print("res : %d\n", res);
```

(continues on next page)

(continued from previous page)

```
  # Now read the same energy through filename
  SysCmd("cp %s_Compound_2.property.txt %s.property.txt", basename, newBasename);
  enFilename.ReadProperty(propertyName=myProperty, filename=newBasename);
  print(" Energy direct    : %.9lf\n", enDirect);
  print(" Energy from file : %.9lf\n", enFilename);
  print("Difference between 2 energies : %.12lf\n", enDirect-enFilename);
End
```

Read_Geom

Read_Geom will read the geometry from a previous step.

Syntax:
Read_Geom *number*

Here number is the number of the job that we want to read the geometry from. The directive should be positioned before a *NewStep* - *StepEnd* block.

Example:

```
#Compound Job 1
New_Step
!BP86 def2-SVP
Step_End

#Compound Job 2
New_Step
!BP86 def2-SVP opt
Step_End

#Compound Job 3
Read_Geom 1
New_Step
!CCSD  def2-SVP
Step_End

End  #Final End
```

In this case the third calculation, through the *Read_MGeom 1* command, will read the geometry from the first calculation.

ReadMOs

ReadMOs reads the molecular orbitals from a previous step.

Syntax:
ReadMOs(stepNumber);

Where:

- *stepNumber:* is the number of the step from which we want to read the orbitals.

Example:

```
# ----------------------------------------------------
# This is an example script for ReadMOs
# ----------------------------------------------------
%Compound
  Variable step = 1;
  New_Step
```

(continues on next page)

(continued from previous page)

```
    !BP86
    *xyzfile 0 1 h2o.xyz
  Step_End

  ReadMOs(step);
  New_Step
    !BP86
  Step_End
End
```

S.GetBasename

In *Compound* strings have all the functionality of a normal variable. In addition they have some additional functions that act only on strings. On of these functions is the function *GetBasename*. This function searches the string and if it contains a dot it will return the part of the string before the dot.

Syntax:

result = source.GetBasename();

Where:

result is the returned string.

source is the original string.

NOTE If the original string contains no dot then the result string will be a copy of the source one.

Example:

```
# ----------------------------------------------------
# This is an example script for string related functions:
#    - GetBasename
#    - GetSuffix
#    - GetChar
# ----------------------------------------------------
%Compound
  Variable original = "lala.xyz";
  Variable basename, suffix;
  Variable constructed = "";
  basename = original.GetBasename();
  suffix   = original.GetSuffix();
  for i from 0 to original.stringlength()-1 Do
    write2String(constructed,"%s%s", constructed, original.GetChar(i));
  endfor
  print("Original    : %s\n", original);
  print("Basename    : %s\n", basename);
  print("Sufix       : %s\n", suffix);
  print("Constructed : %s\n", constructed);
End
```

S.GetChar

In *Compound* strings have all the functionality of a normal variable. In addition they have some additional functions that act only on strings. On of these functions is the function *GetChar*. This function searches the string and if it contains a dot it will return the part of the string after the dot.

Syntax:

result = source.GetChar(index);

Where:

result is the returned string.

index is the index of the curracter in the string. Keep in mind that counting starts with **0** and not 1.

source is the original string.

NOTE If the index is larger than the size of the string or negative then the program will exit.

Example:

```
# -----------------------------------------------------
# This is an example script for string related functions:
#    - GetBasename
#    - GetSuffix
#    - GetChar
# -----------------------------------------------------
%Compound
  Variable original = "lala.xyz";
  Variable basename, suffix;
  Variable constructed = "";
  basename = original.GetBasename();
  suffix   = original.GetSuffix();
  for i from 0 to original.stringlength()-1 Do
    write2String(constructed,"%s%s", constructed, original.GetChar(i));
  endfor
  print("Original    : %s\n", original);
  print("Basename    : %s\n", basename);
  print("Sufix       : %s\n", suffix);
  print("Constructed : %s\n", constructed);
End
```

S.GetSuffix

In *Compound* strings have all the functionality of a normal variable. In addition they have some additional functions that act only on strings. On of these functions is the function *GetSuffix*. This function searches the string and if it contains a dot it will return the part of the string after the dot.

Syntax:

result = source.GetSuffix();

Where:

result is the returned string.

source is the original string.

NOTE If the original string contains no dot then the result string will be an empty string.

Example:

```
# -----------------------------------------------------
# This is an example script for string related functions:
#    - GetBasename
```

(continues on next page)

(continued from previous page)

```
#    - GetSuffix
#    - GetChar
# -----------------------------------------------------
%Compound
  Variable original = "lala.xyz";
  Variable basename, suffix;
  Variable constructed = "";
  basename = original.GetBasename();
  suffix   = original.GetSuffix();
  for i from 0 to original.stringlength()-1 Do
    write2String(constructed,"%s%s", constructed, original.GetChar(i));
  endfor
  print("Original    : %s\n", original);
  print("Basename    : %s\n", basename);
  print("Sufix       : %s\n", suffix);
  print("Constructed : %s\n", constructed);
End
```

StepEnd

StepEnd signals the end of an ORCA Input. It should always be the last directive of an ORCA input inside the compound block that starts with *New_Step* (see paragraph *NewStep*)

NOTE: The old syntax (New_Step/Step_End) is still compatible but please do not use it because in the future it will be deprecated.

Sys_cmd

Sys_cmd will read a system command and execute it.

Syntax:
Sys_cmd *command*

Example:

```
SYS_CMD "orca_mapspc test.out SOCABS -x0700 -x1900 -w0.5 -eV -n10000 "
```

Timer

A *timer* is an object that can keep time for tasks in compound. Before a timer object is used it has to be declared. The declaration of a *timer* is slightly different that the rest of variables, because it has to explicitly declare it's type.

Syntax

timer myTimer;

where

timer is used instead of the normal *variable* command to explicitly set the variable type to *compTimer*.

myTimer is a normal instance of the object.

Example:

```
# ------------------------------------------------
# This is to test timer functions.
# -------------------------------
timer tm;
Variable x = 0.0;
tm.start();
```

(continues on next page)

(continued from previous page)

```
for index from 0 to 100000 Do
  x = x + 0.1;
EndFor
tm.stop();
x = tm.Total();
print( "------------------------------\n");
print( "   Compound - Timer Results   \n");
print( "------------------------------\n");
print( " First total time:   %.2lf\n", x);

x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

Below is a lit of functions that work exclusively on *Timer* objects.

- Last (*T.Last*)
- Reset (*T.Reset*)
- Start (*T.Start*)
- Stop (*T.Stop*)
- Total (*T.Total*)

T.Last

Last is a function that works on a *timer* object. It returns, as a real number, the last value of the timer.

Syntax

myTimer.Last();

Where:

myTimer: is the *timer* object initialized before.

Example:

```
# ----------------------------------------------
# This is to test timer functions.
# --------------------------------
timer tm;
Variable x = 0.0;
tm.start();
for index from 0 to 100000 Do
  x = x + 0.1;
EndFor
tm.stop();
x = tm.Total();
print( "------------------------------\n");
print( "   Compound - Timer Results   \n");
print( "------------------------------\n");
print( " First total time:   %.2lf\n", x);
```

(continues on next page)

(continued from previous page)

```
x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

NOTE Before using last the *timer* object must, beside defined, be also initializee, using *Start* (see *T.Start*)

T.Reset

Reset is a function that works on a *timer* object. It resets the *timer* object to its initial state.

Syntax

myTimer.Reset();

Where:

myTimer: is the *timer* object initialized before.

Example:

```
# ----------------------------------------------
# This is to test timer functions.
# -------------------------------
timer tm;
Variable x = 0.0;
tm.start();
for index from 0 to 100000 Do
  x = x + 0.1;
EndFor
tm.stop();
x = tm.Total();
print( "------------------------------\n");
print( "   Compound - Timer Results   \n");
print( "------------------------------\n");
print( " First total time:   %.2lf\n", x);

x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

T.Start

Start is a function that works on a *timer* object. It retunrs the *timer* object to its initial state.

Syntax

myTimer.Start();

Where:

myTimer: is the *timer* object initialized before.

Example:

```
# ----------------------------------------------
# This is to test timer functions.
# -------------------------------
timer tm;
Variable x = 0.0;
tm.start();
for index from 0 to 100000 Do
  x = x + 0.1;
EndFor
tm.stop();
x = tm.Total();
print( "-----------------------------\n");
print( "   Compound - Timer Results   \n");
print( "-----------------------------\n");
print( " First total time:   %.2lf\n", x);

x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

T.Stop

Stop is a function that works on a *timer* object. It stops the timer from counting.

Syntax

myTimer.Stop();

Where:

myTimer: is the *timer* object initialized before.

Example:

```
# ----------------------------------------------
# This is to test timer functions.
# -------------------------------
timer tm;
Variable x = 0.0;
tm.start();
for index from 0 to 100000 Do
  x = x + 0.1;
```

(continues on next page)

(continued from previous page)

```
EndFor
tm.stop();
x = tm.Total();
print( "------------------------------\n");
print( "   Compound - Timer Results   \n");
print( "------------------------------\n");
print( " First total time:   %.2lf\n", x);

x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

T.Total

Total is a function that works on a *timer* object. It returns a real number with the total time.

Syntax

myTimer.Total();

Where:

myTimer: is the *timer* object initialized before.

Example:

```
# ------------------------------------------------
# This is to test timer functions.
# --------------------------------
timer tm;
Variable x = 0.0;
tm.start();
for index from 0 to 100000 Do
  x = x + 0.1;
EndFor
tm.stop();
x = tm.Total();
print( "------------------------------\n");
print( "   Compound - Timer Results   \n");
print( "------------------------------\n");
print( " First total time:   %.2lf\n", x);

x = tm.total();
tm.Reset();
tm.Start();
for index from 0 to 200000 Do
  x = x + 0.1;
EndFor
tm.Stop();
x = tm.total();

print( " Second total time:   %.2lf\n", x);
End
```

Variables - General

Everything in the *Compound* language is based on variables. Their meaning and usage are similar to those in any programming language: you need to declare a variable and then assign a value to it. Notably, in *Compound*, a variable must be declared before it is assigned a value, following the syntax rules of languages like C. This differs from languages like Python, where you can assign a value to a variable without prior declaration. The only exception to this rule in Compound is the index in a for loop, which does not require prior declaration.

In *Compound* we support the following data types for variables:

- Integer
- Double
- String
- Boolean
- File pointer

In addition to these data types *Compound* supports also variables of type *Geometry* and *Timer* but these are treated separately (see *Geometry* and *Timer*).

For each variable in *Compound* there are 3 major categories of usage:

- The declaration (see *Variables - General*)
- The assignement (see *Variables - Assignment*) and
- Variable functions (see *Variables - Functions*)

Variables - Declaration

There are currently 6 different ways to declare a variable in *Compound*. Their syntax is the following:

Syntax:

A. *Variable name;*

B. *Variable name1, name2;*

C. *Variable name=value;*

D. *Variable name[n];*

E. *Variable name[$n1$][$n2$];*

F. *Variable name={value1, value2, ...}*;

NOTE In previous versions of *Compound* for variables that were matrices but the size was not known one had to declare the variable using the following syntax:

Variable name = [];

This is now changed and the empty brackets are no longer needed, so that this variable can be defined like a normal variable as in case *A*.

Example

```
# ------------------------------------------------------------
# This is to check all available ways of
#   Variable declaration in Compound
# ------------------------------------------------------------
Variable size = 5;
Variable x1;                              #caseA
Variable x2,x3;                           #caseB
Variable x4 = 1.0;                        #caseC
Variable x5 = 0;                          #caseC
```

(continues on next page)

(continued from previous page)

```
Variable x6 = "Test";                      #caseC
Variable x7 = True;                        #caseC
Variable x8 = 2*x4;                        #caseC
Variable x9 = x6;                          #caseC
Variable x10 = x7;                         #caseC
Variable x11;                              #caseA
Variable x12[3];                           #caseD
Variable x12b[size-2];                     #caseD
Variable x12c[size][size-1];               #caseE
x12b[1] = 4.0;
x12c[2][2] = 7.0;
Variable x13[3][3];                        #caseE
Variable x14 = {2.0, 4*x4, 2, "lala"};     #caseF
Variable x15 = {0, 1, 2, 3, 4};
Variable x15b = {0, 1, 2, 3, 4};
Variable x15c[x15[x15b[2]-1]+2];
Variable x16, x17=x10, x18;
Variable x19=2.0, x20, x21[2], x23[size][size];
print( " ---------------------------------------------- \n");
print( " ---------- SUMMARY OF DEFINITIONS ----------- \n");
print( " ---------------------------------------------- \n");
print( " x4 (1.0)          :  %.2lf\n", x4);
print( " x5 (0)            :  %.2d\n",  x5);
print( " x6 (\"Test\")       :  %s\n",    x6);
if (x7) then
  print(" x7 (True)         :  TRUE\n");
else
  print(" x7 (True)         :  FALSE\n");
endIf
print(" x8 (2*x4)         :  %.2lf\n", x8);
print(" x9 (x6)           :  %s\n", x9);
print(" x12b[1] (4.0)     :  %lf\n", x12b[1]);
print(" x12c[2][2] (7.0)  :  %lf\n", x12c[2][2]);
print(" Variable x14 = {2.0, 4*x4, 2, \"lala\"};\n");
print(" x14[0]            :  %lf\n", x14[0]);
print(" x14[1]            :  %lf\n", x14[1]);
print(" x14[2]            :  %d\n",  x14[2]);
print(" x14[3]            :  %s\n",  x14[3]);
print(" Variable x15  = {0, 1, 2, 3, 4};\n");
print(" Variable x15b = {0, 1, 2, 3, 4};\n");
print(" Variable x15c[x15[x15b[2]-1]+2];\n");
print(" x15c.GetSize()    :  %d\n",  x15c.GetSize());
print(" ----------------------------------------------\n");
print(" ----------------------------------------------\n");
End
```

Some comments for the different cases of variable declaration.
Case A is the simplest one were we just declare the name of a variable.

Case B is similar to *Case A* but here more than one variables are declared simultaneously.

In **Case C** we combine the variable declaration with the assignment of a value to the variable. It worth noting that in this case *Compound* automatically deducts the type of the variable based on the given value.

Case D declares a 1-Dimensional array of a defined size.

Case E declares a 2-Dimensional array of defined size. For this case, and also accordingly for *Case E*, one can use previously defined integer variables instead of numbers.

Case F defines an array based on a list of given values. The array will automatically define it's size based on the size of the list. The values in the list do not have to be all of the same type.

NOTE In the past for *Case F* empty brackets were needed after the name of the variable. This is no longer necessary.

NOTE It is important not to forget the final **;** symbol in the end of each declaration because the result of omitting it is undefined.

Variables - Assignment

Assigning a value to a variable has a rather straightforward syntax.

Syntax:
VariableName = CustomFunction;

Where:

VariableName is a variable already declared.

CustomFunction a mathematical expression.

Example:

```
# ------------------------------------------------------------
# This is to check all available ways of variable assignement
#   (It does not take care of 'with' we will have a separate
#         file for this)
# ------------------------------------------------------------

# -----------------------------------
# Some necessary initial declarations
# -----------------------------------
Variable x1, x2, x3, x4;
Variable y1, y2, y3, y4;
Variable x5[4];
Variable y5[4];
Variable x6[3][3];
Variable y6[3][3];

# -----------------------------------
# Now the assignements
# -----------------------------------
#Scalars doubles
x1 = 1.0;
y1 = 2*x1;
#Scalars integers
x2 = 1;
y2 = 2*x2;
#Scalars strings
x3 = "test";
y3 = x3;
#Scalars bools
x4 = True;
y4 = x4;
#1D Arrays
x5[0] = 2.0;
y5[0] = x5[0];
x5[1] = 1;
y5[1] = 2*x5[1];
x5[2] = "test";
y5[2] = x5[2];
x5[3] = True;
y5[3] = x5[3];
#2D Arrays
x6[0][0] = 1.0;
y6[0][0] = 2*x6[0][0];
print( " ---------------------------------------------- \n");
print( " ---------- SUMMARY OF ASSIGNMENTS ---------- \n");
```

(continues on next page)

(continued from previous page)

```
print( " ---------------------------------------------- \n");
print( " ---------------- Scalars -------------------- \n");
print( " x1          :  %.2lf\n", x1);
print( " y1          :  %.2lf\n", y1);
print( " x2          :  %d\n", x2);
print( " y2          :  %d\n", y2);
print( " x3          :  %s\n", x3);
print( " y3          :  %s\n", y3);
print( " ---------------- 1D - Arrays----------------- \n");
print( " x5[0]       :  %.2lf\n", x5[0]);
print( " y5[0]       :  %.2lf\n", y5[0]);
print( " x5[1]       :  %d\n", x5[1]);
print( " y5[1]       :  %d\n", y5[1]);
print( " x5[2]       :  %s\n", x5[2]);
print( " y5[2]       :  %s\n", y5[2]);
print( " ---------------- 2D - Arrays----------------- \n");
print( " x6[0][0]    :  %lf\n", x6[0][0]);
print( " y6[0][0]    :  %lf\n", y6[0][0]);
print(" ----------------------------------------------\n");
print(" ----------------------------------------------\n");
End
```

NOTE It is important to remember to finish the variable assignment using the **';'** symbol.

Variables - Functions

Variables in *Compound* have a small number of functions that can help exctract information about them. In the current version of *Compound* these functions are the following:

Syntax:
VariableName.Function();

where *VariableName* is a variable that is already declared. Then the function will return a value that depends on the *Function* that we used.

Currenlty *Compound* supports the following functions:

- GetBool() *V.GetBool()*
- GetDim1() *V.GetDim1()*
- GetDim2() *V.GetDim2()*
- GetDouble() *V.GetDouble()*
- GetInteger() *V.GetInteger()*
- GetSize() *V.GetSize()*
- GetString() *V.GetString()*
- PrintMatrix() *V.PrintMatrix()*

Example:

```
# ------------------------------------------------------------
# This is to check all available ways of variable assignement
#   (It does not take care of 'with' we will have a separate
#         file for this)
# ------------------------------------------------------------

# -----------------------------------
# Some necessary initial declarations
# -----------------------------------
```

(continues on next page)

(continued from previous page)

```
Variable x1, x2, x3, x4;
Variable y1, y2, y3, y4;
Variable x5[4];
Variable y5[4];
Variable x6[3][3];
Variable y6[3][3];

# ----------------------------------
# Now the assignements
# ----------------------------------
#Scalars doubles
x1 = 1.0;
y1 = 2*x1;
#Scalars integers
x2 = 1;
y2 = 2*x2;
#Scalars strings
x3 = "test";
y3 = x3;
#Scalars bools
x4 = True;
y4 = x4;
#1D Arrays
x5[0] = 2.0;
y5[0] = x5[0];
x5[1] = 1;
y5[1] = 2*x5[1];
x5[2] = "test";
y5[2] = x5[2];
x5[3] = True;
y5[3] = x5[3];
#2D Arrays
x6[0][0] = 1.0;
y6[0][0] = 2*x6[0][0];
print( " ---------------------------------------------- \n");
print( " ---------- SUMMARY OF ASSIGNMENTS ---------- \n");
print( " ---------------------------------------------- \n");
print( " ---------------- Scalars -------------------- \n");
print( " x1         :  %.2lf\n", x1);
print( " y1         :  %.2lf\n", y1);
print( " x2         :  %d\n", x2);
print( " y2         :  %d\n", y2);
print( " x3         :  %s\n", x3);
print( " y3         :  %s\n", y3);
print( " ---------------- 1D - Arrays----------------- \n");
print( " x5[0]      :  %.2lf\n", x5[0]);
print( " y5[0]      :  %.2lf\n", y5[0]);
print( " x5[1]      :  %d\n", x5[1]);
print( " y5[1]      :  %d\n", y5[1]);
print( " x5[2]      :  %s\n", x5[2]);
print( " y5[2]      :  %s\n", y5[2]);
print( " ---------------- 2D - Arrays----------------- \n");
print( " x6[0][0]   :  %lf\n", x6[0][0]);
print( " y6[0][0]   :  %lf\n", y6[0][0]);
print(" ----------------------------------------------\n");
print(" ----------------------------------------------\n");
End
```

V.GetBool()

This function will return a boolean value in case the variable is boolean or integer. For integers it will return *false* for 0 and *true* for all other integer values. In all other cases the program will crash providing a relevant message.

Syntax:

myVar.GetBool();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable double=1.0;
  Variable integer=2;
  Variable iToBool = integer.GetBool();
  Variable boolean=false;

  print("----------------------------------------\n");
  print("     Results for translation functions \n");
  print("Double  to integer : %d    (it should print 1)\n", double.GetInteger());
  print("Integer to double  : %.2lf (it should print 2.00)\n", integer.
↪GetDouble());
  print("Boolean to string  : %s    (it should print FALSE)\n", boolean.
↪GetString());
  print("Integer to boolean : %s    (it should print TRUE)\n", iToBool.
↪GetString());
  print("Double  to string  : %s    (it should print 1.0000000000000000000e+00)\n
↪", double.GetString());
  print("Integer to string  : %s    (it should print 2)\n", integer.GetString());
End
```

V.GetDim1()

This function works on a variable. It will return the size of the first dimension of an array. If the variable is a scalar it will return 1.

Syntax:

myVar.GetDim1();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable dim1, dim2, size;
  Variable A;
  Variable B[3];
  Variable C[3][2];
```

(continues on next page)

(continued from previous page)

```
  print("----------------------------------------\n");
  print("          Results for scalar  \n");
  print("Dim1 : %d (it should print 1)\n", A.GetDim1());
  print("Dim2 : %d (it should print 1)\n", A.GetDim2());
  print("Size : %d (it should print 1)\n", A.GetSize());
  print("----------------------------------------\n");
  print("          Results for 1D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", B.GetDim1());
  print("Dim2 : %d (it should print 1)\n", B.GetDim2());
  print("Size : %d (it should print 3)\n", B.GetSize());
  print("----------------------------------------\n");
  print("          Results for 2D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", C.GetDim1());
  print("Dim2 : %d (it should print 2)\n", C.GetDim2());
  print("Size : %d (it should print 6)\n", C.GetSize());
End
```

V.GetDim2()

This function works on a variable. It will return the size of the second dimension of an array. If the variable is a scalar or a 1-Dimensional array it will return 1.

Syntax:

myVar.GetDim2();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable dim1, dim2, size;
  Variable A;
  Variable B[3];
  Variable C[3][2];

  print("----------------------------------------\n");
  print("          Results for scalar  \n");
  print("Dim1 : %d (it should print 1)\n", A.GetDim1());
  print("Dim2 : %d (it should print 1)\n", A.GetDim2());
  print("Size : %d (it should print 1)\n", A.GetSize());
  print("----------------------------------------\n");
  print("          Results for 1D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", B.GetDim1());
  print("Dim2 : %d (it should print 1)\n", B.GetDim2());
  print("Size : %d (it should print 3)\n", B.GetSize());
  print("----------------------------------------\n");
  print("          Results for 2D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", C.GetDim1());
  print("Dim2 : %d (it should print 2)\n", C.GetDim2());
  print("Size : %d (it should print 6)\n", C.GetSize());
End
```

V.GetDouble()

This function works on a variable. It will return a double value in case the variable is integer or double. In all other cases the program will crash providing a relevant message.

Syntax:

myVar.GetDouble();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable double=1.0;
  Variable integer=2;
  Variable iToBool = integer.GetBool();
  Variable boolean=false;

  print("----------------------------------------\n");
  print("     Results for translation functions \n");
  print("Double  to integer : %d    (it should print 1)\n", double.GetInteger());
  print("Integer to double  : %.2lf (it should print 2.00)\n", integer.
↪GetDouble());
  print("Boolean to string  : %s    (it should print FALSE)\n", boolean.
↪GetString());
  print("Integer to boolean : %s    (it should print TRUE)\n", iToBool.
↪GetString());
  print("Double  to string  : %s    (it should print 1.0000000000000000000e+00)\n
↪", double.GetString());
  print("Integer to string  : %s    (it should print 2)\n", integer.GetString());
End
```

V.GetInteger()

This function works on a variable. It will return an integer value in case the variable is integer or double. In all other cases the program will crash providing a relevant message.

Syntax:

myVar.GetInteger();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable double=1.0;
  Variable integer=2;
  Variable iToBool = integer.GetBool();
  Variable boolean=false;
```

(continues on next page)

(continued from previous page)

```
  print("----------------------------------------\n");
  print("     Results for translation functions \n");
  print("Double  to integer : %d    (it should print 1)\n", double.GetInteger());
  print("Integer to double  : %.2lf (it should print 2.00)\n", integer.
↪GetDouble());
  print("Boolean to string  : %s    (it should print FALSE)\n", boolean.
↪GetString());
  print("Integer to boolean : %s    (it should print TRUE)\n", iToBool.
↪GetString());
  print("Double  to string  : %s    (it should print 1.00000000000000000000e+00)\n
↪", double.GetString());
  print("Integer to string  : %s    (it should print 2)\n", integer.GetString());
End
```

V.GetSize()

This function works on a variable. If the variable is a scalar it will return 1. If the variable is a 1-Dimensional array it will return the size of the array which is the same with the *GetDim1()* . If the variable is a 2-Dimensional array it will return the results Dim1*Dim2.

Syntax:

myVar.GetSize();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable dim1, dim2, size;
  Variable A;
  Variable B[3];
  Variable C[3][2];

  print("----------------------------------------\n");
  print("          Results for scalar  \n");
  print("Dim1 : %d (it should print 1)\n", A.GetDim1());
  print("Dim2 : %d (it should print 1)\n", A.GetDim2());
  print("Size : %d (it should print 1)\n", A.GetSize());
  print("----------------------------------------\n");
  print("          Results for 1D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", B.GetDim1());
  print("Dim2 : %d (it should print 1)\n", B.GetDim2());
  print("Size : %d (it should print 3)\n", B.GetSize());
  print("----------------------------------------\n");
  print("          Results for 2D-Array  \n");
  print("Dim1 : %d (it should print 3)\n", C.GetDim1());
  print("Dim2 : %d (it should print 2)\n", C.GetDim2());
  print("Size : %d (it should print 6)\n", C.GetSize());
End
```

V.GetString()

This function works on a variable. It will return a string of the value of the variable. It works for doubles, integers and booleans.

Syntax:

myVar.GetString();

where:

myVar is an already initialized variable.

Example

```
# ----------------------------------------------------
# This is an example script for
#  Variable functions
# ----------------------------------------------------
%Compound
  Variable double=1.0;
  Variable integer=2;
  Variable iToBool = integer.GetBool();
  Variable boolean=false;

  print("----------------------------------------\n");
  print("     Results for translation functions \n");
  print("Double  to integer : %d    (it should print 1)\n", double.GetInteger());
  print("Integer to double  : %.2lf (it should print 2.00)\n", integer.
↪GetDouble());
  print("Boolean to string  : %s    (it should print FALSE)\n", boolean.
↪GetString());
  print("Integer to boolean : %s    (it should print TRUE)\n", iToBool.
↪GetString());
  print("Double  to string  : %s    (it should print 1.0000000000000000000e+00)\n
↪", double.GetString());
  print("Integer to string  : %s    (it should print 2)\n", integer.GetString());
End
```

V.PrintMatrix()

This function works on variables. It will print print an array on a format with 8 columns.

Syntax: *myVar.PrintMatrix([NCols=numOfColumns]);*

where:

myVar is an already initialized variable.

numOfColumns is the desired number of columns for the printing. This is not obligatory and if not used then by default ORCA will print using **4** columns.

Example

Example:

```
# ---------------------------------------------
#  A script to check PrintMatrix
# ---------------------------------------------
%Compound
  Variable Dim1 = 5;
  Variable Dim2 = 16;
  Variable x[Dim1][Dim2];
  for i from 0 to Dim1-1 Do
```

(continues on next page)

(continued from previous page)

```
    for j from 0 to Dim2-1 Do
      x[i][j] = i+j;
    EndFor;
  EndFor;

  x.PrintMatrix();        # This should print with 4 columns
  x.PrintMatrix(NCols=8); # This should print with 8 columns
EndRun
```

NOTE In case of scalars it will only print the header without any values.

NOTE It only works for arrays of type *'double'* or type *'integer'*. With all variables of other types the program will exit providing an error message.

With

The purpose of the "with" command is to add the ability to call compound while adjusting some of the variables that are already defined in the compound file. This means that if there is a variable defined in the compound file and a value is assigned to it, we can during the call change the assigned value of this variable.

One can pass numbers, string or boolean variables.

It should be noted that it is not possible to call array variables this way. Beside this restriction, the syntax of the variable assignment in the case of with is the same with the variable assignment in a normal *Compound* script.

An important note here is that in case we use the *With* command the *%Compound* block should end with an *'End'* even if we call a *Compound* script file.
Syntax:
% compound *"filename"*
With
var1 = *val1*;
var2 = *val2*;
End
Example:

```
# ------------------------------------------------------------
# This is to check all available ways of variable assignement
#   in combination with the 'with' calls.
# ------------------------------------------------------------

# ---------------------------------
# Some necessary initial definitions
# ---------------------------------
Variable x1, x2, x3, x4;

# ----------------------------------
# Now the assignments
# ----------------------------------
#Scalars doubles
x1 = 1.0;
#Scalars integers
x2 = 1;
#Scalars strings
x3 = "test";
#Scalars bools
x4 = True;
print( " ---------------------------------------------- \n");
print( " ------- SUMMARY OF WITH ASSIGNMENTS --------- \n");
print( " ---------------------------------------------- \n");
print( " The calling input:\n");
```

(continues on next page)

(continued from previous page)

```
print("%Compound \"0975.cmp\"\n");
print("  with\n");
print("    x1       = 3.0;\n");
print("    x2       = 2;\n");
print("    x3       = \"with\";\n");
print("    x4       = False;\n");
print("end\n");
print( " ---------------- Scalars -------------------- \n");
print( " x1 (1.0)      :  %.2lf\n", x1);
print( " x2 (1)        :  %d\n", x2);
print( " x3 (\"test\")   :  %s\n", x3);
print( " x4 (True)     :  %s\n", x4.GetString());
#print( " x6      :  %s\n",    x6);
#if (x4) then
#  print(" x4      : TRUE\n");
#else
#  print(" x4   : FALSE\n");
#endIfo
End
```

Write2File

With the *Print* command (see *Print*) one can write in the ORCA output. Nevertheless it might be that one would prefer to write to a different file. In *Compound* one can achieve this using the *write2File* command. The syntax follows closely the syntax of 'fprintf' command of the programming language C. The arguments definition and the syntax is identical with the syntax of the *Compound 'Print'* command with the addition that one should define a file object to send the printing.

Syntax:
Write2File(file variable, format string, [variables]);

Where:

file variable: is a predefined variable corresponding to an already open, through the *OpenFile* command, file.

format string and *variables* follow exactly the syntax of the *Print* command, so for more details please refere to section *Print*.

NOTE Please remember once everything is writen to the file to close the file, using the *CloseFile* command (see *CloseFile*).

Example:

```
%Compound
  # ------------------------------------------------------------
  #      This is to check all available write2String and
  #                 write2File options
  # ------------------------------------------------------------
  Variable xS    = "test_";
  Variable xI    = 1;
  Variable final;
  Variable fp;
  Variable myFilename = "0955.txt";


  #Create also a file object
  fp = OpenFile(myFilename, "w");
  write2String(final, "   ----- Test ----- \n");
  write2File(fp, "%s", final);
  CloseFile(fp);
```

(continues on next page)

(continued from previous page)

```
  print( " ---------------------------------------------- \n");
  print( " ------ SUMMARY OF WRITE2STRING AND  --------- \n");
  print( " ------          WRITE2FILE          --------- \n");
  print( " ---------------------------------------------- \n");
  write2String(final, "%s", "constant" );
  print( " Final      : %s\n", final);
  write2String(final,"%s", "constant" );      #No space before the quotation marks
  print( " Final      : %s\n", final);
  write2String(final,  "%s", "constant" );    #More than one spaces before
  print( " Final      : %s\n", final);
  write2String(final,"    %s", "constant" );  #No spaces before but  more␣
↪afterwards
  print( " Final      : %s\n", final);
  write2String(final,   "  %s", "constant" ); #More spaces before and more␣
↪afterwards
  print( " Final      : %s\n", final);
  write2String(final, "%s", xS);
  print( " Final      : %s\n", final);
  write2String(final, "%s_%d", xS, xI);
  print( " Final      : %s_%d\n", final, xI);
  write2String(final, "%s_%d", xS,2*xI+1);
  print( " Final      : %s\n", final);
End
```

Write2String

In case one needs to construct a string using some variables, *Compound* provides the *Write2String* command. The syntax of the command is identical with the *Write2File* (see *Write2File*) command with the only exception that instead of a file we should provide the name of a variable that is already declared in the file. The syntax of the format and the variables used is identical with the *Print* command (please refer to *Print*.)

Syntax:

Write2String(variable, format string, [variables]);

where:

variable: is the name of a variable that should already be declared.

format string and *variables* follow exactly the syntax of the *Print* command, so for more details please refer to section *Print*.

Example:

```
%Compound
  # ------------------------------------------------------------
  #      This is to check all available write2String and
  #                 write2File options
  # ------------------------------------------------------------
  Variable xS    = "test_";
  Variable xI    = 1;
  Variable final;
  Variable fp;
  Variable myFilename = "0955.txt";


  #Create also a file object
  fp = OpenFile(myFilename, "w");
  write2String(final, "   ----- Test ----- \n");
  write2File(fp, "%s", final);
  CloseFile(fp);
```

(continues on next page)

(continued from previous page)

```
  print( " ---------------------------------------------- \n");
  print( " ------ SUMMARY OF WRITE2STRING AND  --------- \n");
  print( " ------          WRITE2FILE          --------- \n");
  print( " ---------------------------------------------- \n");
  write2String(final, "%s", "constant" );
  print( " Final      : %s\n", final);
  write2String(final,"%s", "constant" );      #No space before the quotation marks
  print( " Final      : %s\n", final);
  write2String(final,  "%s", "constant" );    #More than one spaces before
  print( " Final      : %s\n", final);
  write2String(final,"    %s", "constant" );  #No spaces before but  more␣
↪afterwards
  print( " Final      : %s\n", final);
  write2String(final,   "  %s", "constant" ); #More spaces before and more␣
↪afterwards
  print( " Final      : %s\n", final);
  write2String(final, "%s", xS);
  print( " Final      : %s\n", final);
  write2String(final, "%s_%d", xS, xI);
  print( " Final      : %s_%d\n", final, xI);
  write2String(final, "%s_%d", xS,2*xI+1);
  print( " Final      : %s\n", final);
End
```

8.3.2 List of known Properties

The name and a sort explanation of all the known variables that can be automatically recovered, from the property file, are given in the next table

Table 8.2: Variables, known to the compound block, with short ex

==========	==========
==========	==========AUTOC
==========	==========
++++++++++++++++++++++++++++++++++++	++++++++++++++++++++++++++++++++++++Energies++++
AUTOCI_REF_ENERGY	AutoCI Reference Energy
AUTOCI_CORR_ENERGY	AutoCI Correlatioin Energy
AUTOCI_TOTAL_ENERGY	AutoCI Total Energy
++++++++++++++++++++++++++++++++++++	++++++++++++++++++++++++++++++++++ ENERGY Gradie
AUTOCI_NUCLEAR_GRADIENT	AutoCI Energy nuclear gradient
AUTOCI_NUCLEAR_GRADIENT_NORM	AutoCI Norm of the nuclear gradient
AUTOCI_NUCLEAR_GRADIENT_ATOM_NUMBERS	AutoCI The atomic numbers of the atoms in the gradient
++++++++++++++++++++++++++++++++++++	++++++++++++++++++++++++ Electric Properties (Dipole mom
AUTOCI_DIPOLE_MAGNITUDE	AutoCI The value of the dipole moment
AUTOCI_DIPOLE_ELEC_CONTRIB	AutoCI The electronic contribution to the dipole moment
AUTOCI_DIPOLE_NUC_CONTRIB	AutoCI The nuclear contribution to the dipole moment
AUTOCI_DIPOLE_TOTAL	AutoCI The total dipole moment
SCF_ENERGY	SCF Energy
++++++++++++++++++++++++++++++++++++	+++++++++++++++++++++++ Electric Properties (Polarizability
AUTOCI_POLAR_ISOTROPIC	AutoCI The polarizability isotropic value
AUTOCI_POLAR_RAW	AutoCI The raw polarizability tensor
AUTOCI_POLAR_DIAG_TENSOR	AutoCI The polarizability diagonalized tensor
AUTOCI_POLAR_ORIENTATION	AutoCI The polarizability orientation (eigenvectors)
++++++++++++++++++++++++++++++++++++	++++++++++++++++++++ Electric Properties (Quadrupole mom

Table 8.2 – continued from previous page

AUTOCI_QUADRUPOLE_MOMENT_ISOTROPIC	AutoCI The quadrupole moment isotropic value
AUTOCI_QUADRUPOLE_MOMENT_DIAG_TENSOR	AutoCI The quadrupole moment diagonalized tensor
AUTOCI_QUADRUPOLE_MOMENT_ELEC_CONTRIB	AutoCI The elctronic contribution to the quadrupole moment tensor
AUTOCI_QUADRUPOLE_MOMENT_NUC_CONTRIB	AutoCI The nuclear contribution to the quadrupole moment tensor
AUTOCI_QUADRUPOLE_MOMENT_TOTAL	AutoCI The total quadrupole moment
++++++++++++++++++++++++++++++++++++++	++++++++++++++++++++++++++ Magnetic Properties (D Tensor
AUTOCI_D_TENSOR_EIGENVALUES	AutoCI The D Tensor eigenvalues
AUTOCI_D_TENSOR_EIGENVECTORS	AutoCI The D Tensor eigenvectors
AUTOCI_D_TENSOR_RAW	AutoCI The Raw D Tensor
AUTOCI_D_TENSOR_D	AutoCI The final D value for the D Tensor
AUTOCI_D_TENSOR_E	AutoCI The final E value for the D Tensor
AUTOCI_D_TENSOR_MULTIPLICITY	AutoCI The spin-multiplicity used for the D Tensor calculation
++++++++++++++++++++++++++++++++++++++	++++++++++++++++++++++++++ Magnetic Properties (G Tensor
AUTOCI_G_TENSOR_RAW	AutoCI The Raw G Tensor
AUTOCI_G_TENSOR_ELEC	AutoCI The Electronic part of the G Tensor
AUTOCI_G_TENSOR_TOT	AutoCI The Total G Tensor
AUTOCI_G_TENSOR_ISO	AutoCI The isotropic g value
AUTOCI_G_TENSOR_ORIENTATION	AutoCI The G Tensor orientation (eigenvectors)
SCF_ENERGY	SCF Energy
VDW_CORRECTION	van der Waals correction
SCF Electric properties	
SCF_DIPOLE_MAGNITUDE.	SCF dipole moment (debye)
SCF_DIPOLE_ELEC_CONTRIB	SCF Electronic contribution to dipole moment
SCF_DIPOLE_NUC_CONTRIB	SCF Nuclear contribution to dipole moment
SCF_DIPOLE_TOTAL	SCF Total dipole moment
SCF_QUADRUPOLE_ISOTROPIC	SCF isotropic quadrupole moment
SCF_QUADRUPOLE_DIAG_TENSOR	SCF quadrupole moment diagonalised tensor
SCF_QUADRUPOLE_ELEC_CONTRIB	SCF electronic contribution to the quadrupole moment
SCF_QUADRUPOLE_NUC_CONTRIB	SCF nuclear contribution to the quadrupole moment
SCF_QUADRUPOLE_TOTAL	SCF total quadrupole moment
SCF_POLAR_ISOTROPIC	SCF isotropic polarizability
SCF_POLAR_RAW	SCF polarizability raw tensor
SCF_POLAR_DIAG_TENSOR	SCF diagonaised polarizability tensor
DBOC Energy Correction	
DBOC_ENERGY	The Diagonal Born-Oppenheimer energy correction
DFT	
DFT_NUM_OF_ALPHA_EL	Number of alpha electrons
DFT_NUM_OF_BETA_EL	Number of beta electrons
DFT_NUM_OF_TOTAL_EL	Total number of electrons
DFT_TOTAL_EN	DFT Total energy
DFT_EXCHANGE_EN	DFT Exchange energy
DFT_CORR_EN	DFT Correlation Energy
DFT_XC_EN	DFT Exchange-Correlation Energy
DFT_NON_LOC_EN	DFT Non-Local correlation
DFT_EMBED_CORR	DFT Embedding correction
gCP correction	
GCP_CORRECTION	gCP energy correction

Table 8.2 – continued from previous page

MP2	
MP2_REF_ENERGY	Reference SCF Energy
MP2_CORR_ENERGY	MP2 Correlation energy
MP2_TOTAL_ENERGY	Total Energy (SCF + MP2)
MP2 Electric properties	
MP2_DIPOLE_MAGNITUDE.	MP2 dipole moment (debye)
MP2_DIPOLE_ELEC_CONTRIB	MP2 Electronic contribution to dipole moment
MP2_DIPOLE_NUC_CONTRIB	MP2 Nuclear contribution to dipole moment
MP2_DIPOLE_TOTAL	MP2 Total dipole moment
MP2_QUADRUPOLE_ISOTROPIC	MP2 isotropic quadrupole moment
MP2_QUADRUPOLE_DIAG_TENSOR	MP2 quadrupole moment diagonalised tensor
MP2_QUADRUPOLE_ELEC_CONTRIB	MP2 electronic contribution to the quadrupole moment
MP2_QUADRUPOLE_NUC_CONTRIB	MP2 nuclear contribution to the quadrupole moment
MP2_QUADRUPOLE_TOTAL	MP2 total quadrupole moment
MP2_POLAR_ISOTROPIC	MP2 isotropic polarizability
MP2_POLAR_RAW	MP2 polarizability raw tensor
MP2_POLAR_DIAG_TENSOR	MP2 diagonaised polarizability tensor
MDCI	
MDCI_REF_ENERGY	Reference SCF Energy
MDCI_CORR_ENERGY	Total Correlation Energy
MDCI_TOTAL_ENERGY	Total Energy (SCF + Correlation)
MDCI_ALPHA_ALPHA_CORR_ENERGY	Correlation energy from $\alpha\alpha$ electron pairs
MDCI_BETA_BETA_CORR_ENERGY	Correlation energy from $\beta\beta$ electron pairs
MDCI_ALPHA_BETA_CORR_ENERGY	Correlation energy from $\alpha\beta$ electron pairs
MDCI_DSINGLET_CORR_ENERGY	Correlation energy from singlet electron pairs (only for closed-shell)(
MDCI_DTRIPLET_CORR_ENERGY	Correlation energy from triplet electron pairs (only for closed-shell) (
MDCI_SSINGLET_CORR_ENERGY	Correlation energy from singlet electron pairs (only for closed-shell) (
MDCI_STRIPLET_CORR_ENERGY	Correlation energy from triplet electron pairs (only for closed-shell) (
MDCI_TRIPLES_ENERGY	Perturbative triples correlation energy
MDCI_ALL_ELECTRONS	Total number of electrons
MDCI_CORR_ELECTRONS	Number of correlated electrons
MDCI_CORR_ALPHA_ELECTRONS	Number of correlated α electrons
MDCI_CORR_BETA_ELECTRONS	Number of correlated β electrons
MDCI Electric properties	
MDCI_DIPOLE_MAGNITUDE	MDCI dipole moment (debye)
MDCI_DIPOLE_ELEC_CONTRIB	MDCI Electronic contribution to dipole moment
MDCI_DIPOLE_NUC_CONTRIB	MDCI Nuclear contribution to dipole moment
MDCI_DIPOLE_TOTAL	MDCI Total dipole moment
MDCI_QUADRUPOLE_ISOTROPIC	MDCI isotropic quadrupole moment
MDCI_QUADRUPOLE_DIAG_TENSOR	MDCI quadrupole moment diagonalised tensor
MDCI_QUADRUPOLE_ELEC_CONTRIB	MDCI electronic contribution to the quadrupole moment
MDCI_QUADRUPOLE_NUC_CONTRIB	MDCI nuclear contribution to the quadrupole moment
MDCI_QUADRUPOLE_TOTAL	MDCI total quadrupole moment
MDCI_POLAR_ISOTROPIC	MDCI isotropic polarizability
MDCI_POLAR_RAW	MDCI polarizability raw tensor
MDCI_POLAR_DIAG_TENSOR	MDCI diagonaised polarizability tensor

Table 8.2 – continued from previous page

CASSCF	
CASSCF_NUM_OF_MULTS	The number of CASSCF spin multiplicities
CASSCF_NUM_OF_IRREPS	The number of CASSCF irreps
CASSCF_FINAL_ENERGY	The CASSCF final energy
PT2_NUM_OF_MULTS	The CASPT2 spin multiplicities
PT2_NUM_OF_IRREPS	The number of CASPT2 irreps
PT2_FINAL_ENERGY	The CASPT2 Energy
DCDCAS_NUM_OF_MULTS	The number of DCDCAS spin multiplicities
DCDCAS_NUM_OF_IRREPS	The number of DCDCAS irreps
DCDCAS_FINAL_ENERGY	The DCDCAS Energy
CASSCF_ABS_SPECTRUM	The CASSCF Absorption spectrum
CASSCF_ABS_SPECTRUM_INFO	Information about the excitations of the CASSCF spectrum
CASSCF_ABS_SPECTRUM_NROOTS	The number of Roots
CASSCF_CD_SPECTRUM	The CASSCF CD spectrum
CASSCF_CD_SPECTRUM_INFO	Information about the excitations of the CASSCF CD spectrum
CASSCF_CD_SPECTRUM_NROOTS	The number or roots
CASPT2_ABS_SPECTRUM	The CASPT2 Absorption spectrum
CASPT2_ABS_SPECTRUM_INFO	Information about the excitations of the CASPT2 spectrum
CASPT2_ABS_SPECTRUM_NROOTS	The number of roots
CASPT2_CD_SPECTRUM	The CASPT2 CD spectrum
CASPT2_CD_SPECTRUM_INFO	Information about the excitations of the CASPT2 CD spectrum
CASPT2_CD_SPECTRUM_NROOTS	The number of roots
CAS_CUSTOM_ABS_SPECTRUM	The Custom CASSCF Absorption spectrum
CAS_CUSTOM_ABS_SPECTRUM_INFO	Information about the excitations of the custom CASSCF absorption
CAS_CUSTOM_ABS_SPECTRUM_NROOTS	The number of roots
CAS_CUSTOM_CD_SPECTRUM	The Custom CASSCF CD spectrum
CAS_CUSTOM_CD_SPECTRUM_INFO	Information about the excitations of the custom CASSCF CD spectru
CAS_CUSTOM_CD_SPECTRUM_NROOTS	The number of roots
DCDCAS_ABS_SPECTRUM	The DCDCAS Absorption spectrum
DCDCAS_ABS_SPECTRUM_INFO	Information about the excitations of the DCDCAS absorption spectru
DCDCAS_ABS_SPECTRUM_NROOTS	The number of roots
CASSCF_DTENSOR_EIGENVALUES	CASSCF D Tensor eigenvalues
CASSCF_DTENSOR_RAW_EIGENVECTORS	CASSCF D Tensor Raw eigenvectors
CASSCF_DTENSOR_D	D value of CASSCF ZFS
CASSCF_DTENSOR_E	E value of CASSCF ZFS
CASSCF_DTENSOR_MULTIPLICITY	Spin multiplicity
CASPT2_DTENSOR_EIGENVALUES	CASPT2 D Tensor eigenvalues
CASPT2_DTENSOR_RAW_EIGENVECTORS	CASPT2 D Tensor raw eigenvectors
CASPT2_DTENSOR_D	D value of CASPT2 ZFS
CASPT2_DTENSOR_E	E value of CASPT2 ZFS
CASPT2_DTENSOR_MULTIPLICITY	Spin multiplicity
CAS_CUSTOM_DTENSOR_EIGENVALUES	custom CASSCF D Tensor eigenvalues
CAS_CUSTOM_DTENSOR_RAW_EIGENVECTORS	custom CASSCF D Tensor Raw eigenvectors
CAS_CUSTOM_DTENSOR_D	D value of custom CASSCF ZFS
CAS_CUSTOM_DTENSOR_E	E value of custom CASSCF ZFS
CAS_CUSTOM_DTENSOR_MULTIPLICITY	Spin multiplicity
CIPSI	
CIPSI_SPIN_MULTIPLICITY	The CIPSI spin multiplicity
CIPSI_NUM_OF_ROOTS	The CIPSI number of roots
CIPSI_FINAL_ENERGY	The CIPSI Final energy
CIPSI_ENERGIES	The CIPSI Energies

Table 8.2 – continued from previous page

CIS	
CIS_FINAL_ENERGY	The final total energy
CIS_ESCF	The SCF Energy
CIS_E0	The Energy of the ground state
CIS_ENERGIES	The singlet energies
CIS_ENERGIESP1	The triplet energies
CIS_MODE	One of the CIS modes
CIS_NUM_OF_ROOTS	The number of roots
CIS_ROOT	State to be optimized
CIS_ABS_SPECTRUM_NROOTS	The number of roots
CIS_ABS_SPECTRUM	The CIS absorption spectrum
CIS_ABS_SPECTRUM_VELOCITY	The CIS absorptioin spectrum in velocity representation
CIS_ABS_SOC_SPECTRUM_NROOTS	The number or roots
CIS_ABS_SOC_SPECTRUM	The CIS absorption spectrum including SOC
CIS_CD_SPECTRUM_NROOTS	The number of roots
CIS_CD_SPECTRUM	The CIS CD spectrum
CIS_CD_SOC_SPECTRUM_NROOTS	The number of roots
CIS_CD_SOC_SPECTRUM	The CIS CD spectrum including SOC
ROCIS	
ROCIS_STATE	ROCIS State
ROCIS_REF_ENERGY	ROCIS Reference energy
ROCIS_CORR_ENERGY	ROCIS correlation energy
ROCIS_TOTAL_ENERGY	ROCIS total energy
ROCIS_ABS_SPECTRUM_NROOTS	Number of roots
ROCIS_ABS_SPECTRUM	ROCIS Absorption spectrum
ROCIS_ABS_SOC_SPECTRUM_NROOTS	Number of roots
ROCIS_ABS_SOC_SPECTRUM	ROCIS absorption spectrum including SOC
ROCIS_CD_SPECTRUM_NROOTS	Number of roots
ROCIS_CD_SPECTRUM	ROCIS CD spectrum
ROCIS_CD_SOC_SPECTRUM_NROOTS	Number of roots
ROCIS_CD_SOC_SPECTRUM	ROCIS CD spectrum including SOC
MRCI	
MRCI_ABS_SPECTRUM	The MRCI absorption spectrum
MRCI_ABS_SPECTRUM_INFO	Information about the absorption spectrum
MRCI_ABS_SPECTRUM_NROOTS	The number of roots
MRCI_CD_SPECTRUM	The MRCI CD spectrum
MRCI_CD_SPECTRUM_INFO	Information about the MRCI CD spectrum
MRCI_CD_SPECTRUM_NROOTS	The number of roots
MRCI_DIPOLE_MOMENTS	The MRCI dipole moments
MRCI_DIPOLE_MOMENTS_INFO	Information about the MRCI dipole moments
MRCI_DTENSOR_EIGENVECTORS	The eigenvectors of the MRCI D tensor
MRCI_DTENSOR_EIGENVALUES	The eigenvalues of the MRCI D tensor
MRCI_DTENSOR_RAW_EIGENVECTORS	The raw eigenvectors of the MRCI D tensor
MRCI_DTENSOR_D	The MRCI D value for the ZFS
MRCI_DTENSOR_E	The MRCI E value for the ZFS
MRCI_DTENSOR_MULTIPLICITY	The MRCI spin multiplicity
EXTRAPOLATION	
EXTRAP_SCF_ENERGIES	The SCF energies with the different basis sets

Table 8.2 – continued from previous page

EXTRAP_CBS_SCF	The extrapolated SCF energy
EXTRAP_CORR_ENERGIES	The correlation energies with the different basis sets
EXTRAP_CBS_CORR	The extrapolated correlatioin energy
EXTRAP_CBS_TOTAL	The extrapolated total energy
EXTRAP_CCSDT_X	The (T) contribution to the energy
EXTRAP_NUM_OF_ENERGIES	The number of energies (basis sets) used for the extrapolation
THERMOCHEMISTRY	
THERMO_TEMPERATURE	Temperature (^oK)
THERMO_PRESSURE	Pressure (Atm)
THERMO_TOTAL_MASS	Total Mass of the molecule (AMU)
THERMO_SPIN_DEGENERACY	Electronic degeneracy
THERMO_ELEC_ENERGY	Electronic energy (Eh)
THERMO_TRANS_ENERGY	Translational energy (Eh)
THERMO_ROT_ENERGY	Rotational energy (Eh)
THERMO_VIB_ENERGY	Vibrational energy (Eh)
THERMO_NUM_OF_FREQS	The number of vibrational frequencies
THERMO_FREQS	Frequencies
THERMO_ZPE	Zero point energy (Eh)
THERMO_INNER_ENERGY_U	Inner Energy (Eh)
THERMO_ENTHALPY_H	Enthalpy (Eh)
THERMO_ELEC_ENTROPY	(Electronic Entropy)*T (Eh)
THERMO_ROR_ENTROPY	(Rotational Entropy)*T (Eh)
THERMO_VIB_ENTROPY	(Vibrational Entropy)*T (Eh)
THERMO_TRANS_ENTROPY	(Translational Entropy)*T (Eh)
THERMO_ENTROPY_S	(Total Entropy)*T (Eh)
THERMO_FREE_ENERGY_G	Free Energy (Eh)
EPR-NPR Spin-Spin coupling	
EPRNMR_SSC_NUM_OF_NUC_PAIRS	Number of nuclear pairs to calculate something
EPRNMR_SSC_NUM_OF_NUC_PAIRS_DSO	Number of nuclear pairs to calculate DSO terms
EPRNMR_SSC_NUM_OF_NUC_PAIRS_PSO	Number of nuclear pairs to calculate PSO terms
EPRNMR_SSC_NUM_OF_NUC_PAIRS_FC	Number of nuclear pairs to calculate FC terms
EPRNMR_SSC_NUM_OF_NUC_PAIRS_SD	Number of nuclear pairs to calculate SD terms
EPRNMR_SSC_NUM_OF_NUC_PAIRS_SD_FC	Number of nuclear pairs to calculate SD/FC terms
EPRNMR_SSC_NUM_OF_NUCLEI_PSO	Number of nuclei to calculate PSO perturbations
EPRNMR_SSC_NUM_OF_NUCLEI_FC	Number of nuclei to calculate SD/FC perturbations
SOC Energy Correction	
SOC_NUCLEAR_ENERGY	The nuclear energy
SOC_2C_ENERGY	The total 2-component energy
SOC_NON_SOC_ENERGY	The non-SOC total energy
SOC_ENERGY_CORRECTION	The SOC energy correction
Solvation	
SOLVATION_EPSILON	Dielectric constant
SOLVATION_REFRAC	Refractive index
SOLVATION_RSOLV	Solvent probe radius
SOLVATION_SURFACE_TYPE	Cavity surface
SOLVATION_CPCM_DIEL_ENERGY	Total energy including the CPCM dielectric correction
SOLVATION_NPOINTS	Number of points for the Gaussian surface

Table 8.2 – continued from previous page

SOLVATION_SURFACE_AREA	Surface area
General Job Information	
JOB_INFO_MULT	Job Multiplicity
JOB_INFO_CHARGE	Job Charge
JOB_INFO_NUM_OF_ATOMS	Total number of atoms
JOB_INFO_NUM_OF_EL	Total number of electrons
JOB_INFO_NUM_OF_FC_EL	Number of frozen core electrons
JOB_INFO_NUM_OF_CORR_ELC	Number of correlated electrons
JOB_INFO_NUM_OF_BASIS_FUNCS	Number of basis functions
JOB_INFO_NUM_OF_AUXC_BASIS_FUNCS	Number of auxilliary C basis functions
JOB_INFO_NUM_OF_AUXJK_BASIS_FUNCS	Number of auxilliary J basis functions
JOB_INFO_NUM_OF_AUX_CABS_BASIS_FUNCS	Number of auxilliary JK basis functions
JOB_INFO_NUM_OF_AUX_CABS_BASIS_FUNCS	Number of auxilliary CABS basis functions
JOB_INFO_TOTAL_EN	Final energy
HESSIAN	
HESSIAN_MODES	The hessian
Math Functions	
ABS	Absolute value
COS	Cosine
SIN	Sine
TAN	Tangent
ACOS	Inverse cosine
ASIN	Inverse sine
ATAN	Inverse tangent
COSH	Hyperbolic cosine
SINH	Hyperbolic sine
TANH	Hyperbolic tangent
EXP	Exponential
LOG	Common logarithm
LN	Natural logarithm
SQRT	Square root
ROUND	Round down to nearest integer

8.4 Compound Examples

8.4.1 Introduction

A library of compound scripts exist in page https://github.com/ORCAQuantumChemistry/CompoundScripts .

8.4.2 Hello World

Introduction

This is the simplest script that nevertheless points to an important feature of *Compound*. That is the fact that *Compound* does not have to run an actual 'normal' ORCA calculation but it can also be used as a driver for various tasks, in this case to just print a message.

Filename

helloWorld.inp

SCRIPT

```
%Compound
  print("Hellow World!\n");
EndRun
```

8.4.3 New Job

Introduction

One of the features of ORCA that will be deprecated in the future and should not be used any more is the *'New_Job'* feature. The current script is a simple example how *Compound* can be used to just run a series of calculations.

Filename

replaceNewJob.inp

SCRIPT

```
# This is a small script thas shows how
# 'Compound' can replace the previous
# ORCA '$New_Job' feature
%Compound
  # -----------------------------------
  # First job
  # -----------------------------------
  New_Step
    !BP86
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
  # -----------------------------------
  # Second job with same goemetry
  # but different functional
  # -----------------------------------
  New_Step
    !B3LYP
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
EndRun
```

Comments

From the *Compound* point of view the syntax in this script is not the most efficient one. It can be rewritten in more compact, cleaner, general way. Neverteless this is meant only as an exmample of how *Compound* can replace older ORCA calculations that used the, to be deprecated, *'New_Job'* feature.

8.4.4 High Accuracy

Introduction

This is a script that utilizes the scheme by N. J. DeYonker, T. R. Cundari, and A. K. Wilson published on: J. Chem. Phys. 124, 114104 (2006). The script calculates accurate total energies of molecules.

Filename

ccCA_CBS_2.cmp

SCRIPT

```
# This is a small script thas shows how
# 'Compound' can replace the previous
# ORCA '$New_Job' feature
%Compound
  # -----------------------------------
  # First job
  # -----------------------------------
  New_Step
    !BP86
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
  # -----------------------------------
  # Second job with same goemetry
  # but different functional
  # -----------------------------------
  New_Step
    !B3LYP
    *xyz 0 1
      H 0.0 0.0 0.0
      H 0.0 0.0 0.8
    *
  Step_End
EndRun
```

Comments

It is interesting that in this scheme the total energy is treated and there is not separation in extrapolation between HF energy and correlation energy.

8.4.5 Scan

Introduction

This is an example script for a 1-Dimensional geometry scan. It is set up for the Ne-Ne bond distance but can be modified to suit the user's specific needs.

Filename

scan_1D_1M_1P.cmp

SCRIPT

```
# Author : Dimitrios G. Liakos
# Date   : May of 2024
#
# This is a script that will calculate and potentially
#   plot ONE property(1P) along a scan in ONE dimesion (1D)
#   using only ONE method (1M)
#
#   It is part of a series of scripts for different
#     combinations of scans for dimensions, methods,
#     and properties
#
# Here as an example we use for:
#    - dimension: the Ne-Ne bond (dist)
#    - method   : "HF" (method)
#    - property : the SCF energy (propName)
#
# The script creates a csv file with the absolute energies
#   and an additional one with the potential energies  in
#   kcal/mol. Both will be saved on disk.
#
# If 'DoPython' is set to true it will also create a python
#   script that plots the generated values and then run
#   it. The python script will be saved on disk and thus one
#   can afterwards manipulate it.
#
# NOTE The boolean option plotPotential will choose between
#      plotting absolute values or potential.
#
# NOTE The boolean obtion doKcal if set to true multiplies
#      the potential values with the HartreeToKcal factor.
#
# NOTE In case the doPython is set to true the script expects
#   that python3 is avaiable and also the following libraries:
#   - pandas
#   - seaborn
#   - matplotlib.pyplot
#
# ------------------------------------------------------------
#
# ----------------    Variables to change (e.g. through 'with')   -----------------
↪---------
Variable method       = "HF";             # The methods of the calculation
Variable basis        = "cc-pVDZ";        # The basis set of the calculation
Variable restOfInput  = "TightSCF";       # Maybe something common for the simple␣
↪input
Variable charge       = 0;                # Charge
Variable mult         = 1;                # Spin multiplicity
Variable myPropName   = "SCF_Energy";     # The properties we want to read
#
Variable lowerLimit   = 2.5;              # Lower limit value
Variable UpperLimit   = 5.0;              # Upper limit value
```

(continues on next page)

(continued from previous page)

```
Variable NSteps       = 13;                # Number of steps for the grid
Variable baseFilename = "myPotential";     # The basename for the created files
Variable plotPotential= true;              # Plot the potential instead of absolute␣
↪values
Variable DoKcal       = true;              # Multiply the potential values with the␣
↪HartreeToKcal factor
Variable removeFiles  = true;              # Remove *_Compound_*, *bas* files
# ------------------  python plot relevant variables ----------------------------
↪---------
Variable DoPython     = true;              # if we want python or not
Variable lw           = 4;                 # The line width in case we plot with␣
↪python
Variable marker       = "o";               # The type of markers
Variable markerSize   = 10;                # The size of the markers in case we plot
Variable fontSize     = 18;
#
# ------------------------      Rest of the variables       ------------------------
↪---------
#
Variable HartreeToKcal = 627.5096080305927;                    # Hartree to kcal/
↪mol conversion factor
Variable stepSize      = (UpperLimit-LowerLimit)/(NSteps-1);  # The stepsize of␣
↪the grid
Variable calcValues[NSteps];                                   # An array to store␣
↪the calculated values
Variable  res, dist, calcValue;
Variable myFilename, csvFilename;
Variable fPtr;                                                 # A file to write

# ---------------------------------------------------
#  Open and Write file header for the absolute values
# ---------------------------------------------------
write2String(csvFilename, "%s_absValues.csv", baseFilename);
fPtr = OpenFile(csvFilename, "w");
write2File(fptr, "distance,method,property,calcValue\n");

# ---------------------------------------------------
#  Perform the calculations and update the file
# ---------------------------------------------------
for iStep from 0 to NSteps-1 Do
  dist = lowerLimit + (iStep)*stepSize;
  New_Step
    !&{method} &{basis} &{restOfInput}
    *xyz &{charge} &{mult}
      Ne 0.0 0.0 0.0
      Ne 0.0 0.0 &{dist}
    *
    Step_end
    res = calcValue.readProperty(propertyName=myPropName);
    write2File(fPtr, "%.4lf,%20s,%20s,%20.10lf\n", dist, method,myPropName,␣
↪calcValue);
    calcValues[iStep]=calcValue;
EndFor
CloseFile(fPtr);   # Close the file

# ---------------------------------------------------
# Evaluate and write the relative values
# ---------------------------------------------------
write2String(csvFilename, "%s_relValues.csv", baseFilename);
fPtr = OpenFile(csvFilename, "w");
write2File(fPtr, "distance,method,property,calcValue\n");
```

(continues on next page)

(continued from previous page)

```
for iStep from 0 to NSteps-1 Do
  dist = lowerLimit + (iStep)*stepSize;
  if (DoKcal) then
    calcValue = (calcValues[iStep]-calcValues[NSteps-1])*HartreeToKcal;
  else
    calcValue = calcValues[iStep]-calcValues[NSteps-1];
  EndIf
  write2File(fPtr, "%.4lf,%20s,%20s,%20.10lf\n", dist, method,myPropName,␣
↪calcValue);
EndFor
CloseFile(fPtr);   # Close the file

if (removeFiles) then
  sys_cmd("rm *_Compound_* *.bas*");
EndIf

# ----------------------------------------------------
#  Create a python file and run it
# ----------------------------------------------------
if (DoPython) then
  if (plotPotential) then
    write2String(csvFilename, "%s_relValues.csv", baseFilename);
  else
    write2String(csvFilename, "%s_absValues.csv", baseFilename);
  endIf

  write2String(myFilename, "%s.py", baseFilename);
  fPtr = openFile(myFilename, "w");
  # Import necessary libraries
  write2File(fPtr, "import pandas as pd\n");
  write2File(fPtr, "import seaborn as sns\n");
  write2File(fPtr, "import matplotlib.pyplot as plt\n");
  # Read the csv file
  write2File(fPtr, "df = pd.read_csv('%s')\n", csvFilename);
  #Make a lineplot
  write2File(fPtr, "sns.lineplot(data=df, x=\"distance\", y=\"calcValue\", hue=\
↪"property\", \n
                      lw=%d, markers=True, marker='%s', markersize=%d,␣
↪dashes=False)\n", lw, marker, markersize);
  write2File(fPtr, "plt.axhline(y=0, color='black', linestyle='-', linewidth=1)\n
↪");
  write2File(fPtr, "plt.title(\"Energy Potential\", fontsize=%d)\n", fontsize+4);
  write2File(fPtr, "plt.xlabel(\"Ne-Ne Distance\", fontsize=%d)\n", fontsize);
  write2File(fPtr, "plt.ylabel(\"Energy (kcal/mol)\", fontsize=%d)\n", fontsize);
  write2File(fPtr, "plt.xticks(fontsize=%d)\n", fontSize);
  write2File(fPtr, "plt.yticks(fontsize=%d)\n", fontSize);
  write2File(fPtr, "plt.show()\n");
  closeFile(fPtr);
  sys_cmd("python3 %s", myFilename);
EndIf

End
```

Comments

This script has some interesing features. It contains two variables *removeFiles* and *DoPython*. If the first of them is set to *true* then the script will use a system command to remove files that are not needed anymore after the end of the calculation. The latter, *DoPython*, if set to *true* will read the *.csv* file that is created and write a *python* file to make a plot of the results. Then it will run the python script to actually make the plot.

8.4.6 Numerical polarizabilities

Introduction

This script calculates numerically the polarizability of the molecule using single point calculations with an electric field.

Filename

numericalPolarizability.cmp

SCRIPT

```
# Authors: Dimitrios G. Liakos / Frank Neese / Zikuan Wang
# Date    : May of 2024
#
# This is a compound script that calculates the
#  dipole-dipole polarizability tensor numerically
#  using the double derivative of energy.
#
# The idea is the following:
#
# 1 Perform a field free calculation
#
# 2 Loop over directions I=X,Y,Z
#
# 3 Loop over directions J=X,Y,Z
#
#    - put a small Q-field in directions I and J
#    - Solve equations to get the energy for each combination
#    - Polarizability alpha(I,J) =- ( E(+I,+J) - E(+I,-J)-E(-I,+J)+ E(-i,-j)/
↪(4*Field^2)
# 4 Print polarisability
#
# ---------------------------------------------------------------------
# ----------------------    Variables    -------------------------------
# --- Variables to be adjusted (e.g. using 'with' ----------------------
Variable molecule    = "h2o.xyz";
Variable charge      = 0;
Variable mult        = 1;
Variable method      = "HF";
Variable basis       = " ";
Variable restOfInput = "VeryTightSCF";
Variable blocksInput = " ";
Variable E_Field     = 0.0001;
Variable enPropName  = "JOB_Info_Total_En";
Variable removeFiles = true;
# -------------- Rest of the variables --------------------------------
Variable FField[3];
Variable EFree, EPlusPlus, EPlusMinus, EMinusPlus, EMinusMinus, a[3][3];
Variable FFieldStringPlusPlus, FFieldStringPlusMinus;
Variable FFieldStringMinusPlus, FFieldStringMinusMinus;
Variable aEigenValues, aEigenVectors;

# ----------------------------------------
# Calculation without field
# ----------------------------------------
New_Step
  !&{method} &{basis} &{restOfInput}
  &{blocksInput}
  *xyzfile &{charge} &{mult} &{molecule}
Step_End
EFree.ReadProperty(propertyName=enPropName);
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------------------
# Loop over the x, y, z directions
# ------------------------------------------------------------
for i from 0 to 2 Do
  for j from 0 to 2 Do
    # ------------------------------------------------------------
    # Create the appropriate direction oriented field string
    # ------------------------------------------------------------
    # ---------------------- (++) ------------------------------
    for k from 0 to 2 Do
      FField[k] = 0.0;
    EndFor
    FField[i] = FField[i] + E_Field;
    FField[j] = FField[j] + E_Field;
    write2String(FFieldStringPlusPlus,   " %lf,  %lf,  %lf",
    FField[0], FField[1], FField[2]);
    #
    # --------------------- (+-) -----------------------------
    for k from 0 to 2 Do
      FField[k] = 0.0;
    EndFor
    FField[i] = FField[i] + E_Field;
    FField[j] = FField[j] - E_Field;
    write2String(FFieldStringPlusMinus,   " %lf,  %lf,  %lf",
    FField[0], FField[1], FField[2]);
    #
    # --------------------- (-+) -----------------------------
    for k from 0 to 2 Do
      FField[k] = 0.0;
    EndFor
    FField[i] = FField[i] - E_Field;
    FField[j] = FField[j] + E_Field;
    write2String(FFieldStringMinusPlus,   " %lf,  %lf,  %lf",
    FField[0], FField[1], FField[2]);
    #
    # --------------------- (--) -----------------------------
    for k from 0 to 2 Do
      FField[k] = 0.0;
    EndFor
    FField[i] = FField[i] - E_Field;
    FField[j] = FField[j] - E_Field;
    write2String(FFieldStringMinusMinus,   " %lf,  %lf,  %lf",
    FField[0], FField[1], FField[2]);

    # ----------------------------------------
    # Perform the calculations.
    # The plus_plus (++) one
    # ----------------------------------------
    ReadMOs(1);
    New_Step
      !&{method} &{basis} &{restOfInput}
      %SCF
        EField = &{FFieldStringPlusPlus}
      End
      &{blocksInput}
    Step_End
    EPlusPlus.readProperty(propertyName=enPropName);
    # ----------------------------------------
    # The plus_minus (+-) one
    # ----------------------------------------
    ReadMOs(1);
```

(continues on next page)

(continued from previous page)

```
    New_Step
      !&{method} &{basis} &{restOfInput}
       %SCF
        EField = &{FFieldStringPlusMinus}
      End
      &{blocksInput}
    Step_End
    EPlusMinus.readProperty(propertyName=enPropName);
    # ----------------------------------------
    # The minus_plus (-+) one
    # ----------------------------------------
    ReadMOs(1);
    New_Step
      !&{method} &{basis} &{restOfInput}
      %SCF
        EField = &{FFieldStringMinusPlus}
      End
      &{blocksInput}
    Step_End
    EMinusPlus.readProperty(propertyName=enPropName);
    # ----------------------------------------
    # And the minus_minus (--) one
    # ----------------------------------------
    ReadMOs(1);
    New_Step
      !&{method} &{basis} &{restOfInput}
       %SCF
        EField = &{FFieldStringMinusMinus}
      End
      &{blocksInput}
    Step_End
    EMinusMinus.readProperty(propertyName=enPropName);

    a[i][j] = -(EPlusPlus-EPlusMinus-EMinusPlus+EMinusMinus)/(4*E_Field*E_Field);
  EndFor
EndFor

# ----------------------------------------
# Diagonalize
# ----------------------------------------
a.Diagonalize(aEigenValues, aEigenVectors);

# ----------------------------------------
# Do some printing
# ----------------------------------------
print( "\n\n");
print( " -------------------------------------------------------\n");
print( "                   COMPOUND                            \n");
print( " Numerical calculation of dipole polarizability\n");
print( " -------------------------------------------------------\n");
print( " Molecule    : %s\n", molecule);
print( " charge      : %d\n", charge);
print( " Mult        : %d\n", mult);
print( " Method      : %s\n", method);
print( " Basis       : %s\n", basis);
print( " RestOfInput : %s\n", restOfInput);
print( " BlocksInput : %s\n", blocksInput);
print( " The electric field perturbation used was:    %.5lf a.u.\n", E_Field);
print( " \n\n");

print( " -------------------------------------------------------\n");
```

(continues on next page)

(continued from previous page)

```
print( " Raw electric dipole  polarizability tensor is:\n");
print( " -------------------------------------------------------\n");
For i from 0 to 2 Do
  print("%13.8lf  %13.8lf  %13.8lf\n", a[i][0], a[i][1], a[i][2]);
EndFor
print( " -------------------------------------------------------\n");
print("\n");

print( " -------------------------------------------------------\n");
print( " Raw electric dipole polarizability Eigenvalues\n");
print( " -------------------------------------------------------\n");
print("%13.8lf  %13.8lf  %13.8lf\n", aEigenValues[0], aEigenValues[1],␣
↪aEigenValues[2]);
print( " -------------------------------------------------------\n");
print("\n");

print( " -------------------------------------------------------\n");
print( " Raw electric dipole polarizability Eigenvectors\n");
print( " -------------------------------------------------------\n");
For i from 0 to 2 Do
  print("%13.8lf  %13.8lf  %13.8lf\n", aEigenVectors[i][0], aEigenVectors[i][1],␣
↪aEigenVectors[i][2]);
EndFor

print( "\n a isotropic value : %.5lf\n",␣
↪(aEigenValues[0]+aEigenValues[1]+aEigenValues[2])/3.0);
print( " -------------------------------------------------------\n");
print("\n\n");
#
#
# ---------------------------------------------------
#  Maybe remove unneccesary files
# ---------------------------------------------------
if (removeFiles) then
  sys_cmd("rm *_Compound_* *.bas* ");
EndIf
#
End
```

Comments

In this script we also use the linear algebra *diagonalize* function that is available in *Compound*.

8.4.7 Iterative optimization

Introduction

This is a script that will perform a geometry optimization, then run a frequency calculation and in case there are negative frequencies it will adjust the geometry, based on the Hessian, and optimize again.

Filename

iterativeOptimization.cmp

SCRIPT

```
# Author: Dimitrios G. Liakos and Franke Neese
# Date  : May/June of 2024
#
# ************************************** DESCRIPTION␣
↪*********************************************
```

(continues on next page)

(continued from previous page)

```
# iterative Optimization protocol to find structure with no negative
# frequencies (e.g. real minima)
#
# Step 1. Run a single point calculation (we need it for the first property file)
#
# Step 1. Loop and perform calculations with (optimization and frequencies)
#
# Step 2. Check the frequencies. If there are negative ones use the hessian
#         of the appropriate normal mode to adjust the geometry
#
# ------ Variables to adjust (e.g. using 'with') -------------------
Variable method          = "HF"; #"HF-3c";
Variable MaxNTries       =  25;   # Number of maximum tries
Variable CutOff          = -10.0; # CutOff for a negative frequency
Variable scaling         = 0.6;   # Scaling factor for normal mode
Variable NNegativeTarget = 0;     # Number of negative frequencies we allow
Variable myFilename      = "xyzInput.xyz";
Variable charge          = 0;
Variable multiplicity    = 2;
# ----------------------------------------------------------------
# ------            Rest of variables            -------------------
Geometry myGeom;
Variable freqs, modes;
Variable res = -1;
Variable NNegative =   0;
Variable OptDone;

# ---------------------------------------------------------
# Perform a single point calculation. We need it for
# the initial geometry from the property file
# ---------------------------------------------------------
New_Step
  !&{method}
Step_End
myGeom.Read();
myGeom.WriteXYZFile(filename=myFilename);

# ---------------------------------------------------------
# Start a for loop over number of tries
# ---------------------------------------------------------
For itry From 1 To maxNTries Do
  # ---------------------------------------------
  # Perform a geometry optimization/Frequency calculation
  # ---------------------------------------------
  New_Step
    ! &{method} freq Opt
    *xyzfile &{charge} &{multiplicity} &{myFilename}
  Step_End
  res = freqs.readProperty(propertyName = "THERMO_FREQS");
  res = modes.readProperty(propertyName = "HESSIAN_MODES");
  myGeom.Read();

  # ---------------------------------------------
  #  check for sufficiently negative frequencies
  # ---------------------------------------------
  NNegative = 0;
  For ifreq From 0 to freqs.GetSize()-1 Do
    if ( freqs[ifreq] < CutOff )  then
       myGeom.FollowNormalMode(vibrationSN=ifreq, scalingFactor=scaling);
       NNegative = NNegative + 1;
    endif
```

(continues on next page)

(continued from previous page)

```
  endfor
  myGeom.WriteXYZFile(filename=myFilename);
  If ( NNegative <= NNegativeTarget ) then
    goto OptDone;
  endif
endfor

# ----------------------------------------------------------------
# Either found correct geometry or reached maximum number of tries.
# ----------------------------------------------------------------
OptDone :
if (NNegative > NNegativeTarget) then
  print("ERROR The program did not find a structure with the desired\n number of␣
↪imaginary frequencies.\n There are %9.3lf negative frequencies after %3d steps",␣
↪NNegative,itry);
else
  print("\nSUCCESS optimized structure with (%d) negative\n frequencies found␣
↪after %3d steps", NNegative, itry);
endif

End
```

8.4.8 Gradient extrapolation

Introduction

This script extrapolates the gradient of a molecule. It uses a two point extrapolation where the Hartree-Fock and correlation parts of the gradient are extrapolated separately. This opens the way for geometry optimizations with extrapolated gradients.

Filename

gradientExtrapolation.cmp

SCRIPT

```
#   Author: Dimitrios G. Liakos and Frank Neese
#   Date  : May of 2024
#
#   This is a compound file that extrapolates the
#   energy gradients to Complete Basis Set Limit (CBS).
#
#   STEPS:
#   Step1 : Run HF calculation with small basis set
#           Read scfGradX and scfEnX
#   Step2 : Run Correlation calculation with small basis set
#           Read totalGradX and totalEnX
#   Step3 : Calculate the gradient differene to get
#           the corrGradX (only the correlation part)
#   Step4 : Run HF calculation with big basis set
#           Read scfGradY and scfEnY
#   Step5 : Run correlation calculation with big basis set
#           Read totalGradY and totalEnY
#   Step6 : Calculate the gradient difference with the
#           big basis set to get corrGradY
#   Step7 : Evaluate scfGradCBS and scfEnCBS
#           using scfGradX and scfGradY
#   Step8 : Evaluate corrGradCBS using
#           corrGradX and corrGradY
#   Step9 : Add scfGradCBS and corrGradCBS to get
#           totalGradCBS
```

(continues on next page)

(continued from previous page)

```
#   Step10: If needed, create an ORCA engrad file
#
#
#   NOTE: It works with an xyz file the name of which we should provide.
#         using the variable initialXYZFilename.
#
#   We extrapolate the SCF part using the scheme
#     proposed in: J. Phys. Chem. 129, 184116, 2008
#       E_SCF(X) = E_SCF(CBS)+Aexp(-a*SQRT(X))
#
#   We extrapolate the correlation part using the schem
#     proposed in: J. Chem. Phys. 1997, 106, 9639
#   E_CBS(CORR)=(X^b*E_X(CORR)-Y^b*E_Y(CORR))/(X^b-Y^b)

#   We use alpha and beta exponents proposed in:
#     J. Chem. Theory Comput., 7, 33-43 (2011)
# -----------------------    Variables    ------------------------------
# --- Variables to be adjusted (e.g. using 'with' -----------------------
Variable Molecule           = "initial.xyz";   # xyz file of the initial structure
Variable charge             = 0;               # Charge
Variable multiplicity       = 1;               # Spin multiplicity
Variable method             = "MP2";           # The method we use for the␣
↪calculation
Variable LowerBasis         = "cc-pVDZ";       # Small basis set
Variable UpperBasis         = "cc-pVTZ";       # Big basis set
Variable restOfInput        = "EnGrad ";       # The rest of the simple input
Variable addCorrelation     = true;            # If we have a correlation part
Variable scfEnPropName      = "MP2_Ref_Energy";  # The name of the property for␣
↪the SCFenergy
Variable corrEnPropName     = "MP2_Corr_Energy"; # The name of the property for␣
↪the correlation energy
Variable LowerCardinal      = 2;               # Cardinal number of small basis set
Variable UpperCardinal      = 3;               # Cardinal number of big basis set
Variable alpha              = 4.420;           # Exponent for SCF extrapolation
Variable beta               = 2.460;           # Exponent for corrleation␣
↪extrapolation
Variable enGradFilename     = "result.engrad"; # Filename of the ORCA engrad file
Variable produceEnGradFile  = true;            # Produce an ORCA engrad file
# ---------------------------------------------------------------------
# -------------- Rest of the variables -------------------------------
Geometry myGeom;
Variable scfGradX, scfGradY;                   # SCF Gradients
Variable scfEnX, scfEnY, scfEnCBS;             # SCF energies
Variable corrEnX, corrEnY, corrEnCBS;          # Correlation enegies
Variable totalGradX, totalGradY;               # Total Gradients
Variable eX = 0.0;
Variable eY = 0.0;
Variable res = -1;

Variable denominator = 0.0;
Variable gradX = 0.0, gradY = 0.0, gradCBS=0.0;
Variable nAtoms = 0;
Variable EnGradFile;
Variable Cartesians, AtomicNumbers;

# ---------------------------------------------------------------------
# Step 1. SCF Calculation with small basis set (X)
# ---------------------------------------------------------------------
New_Step
  ! HF &{LowerBasis} &{restOfInput}
  *xyzfile &{charge} &{multiplicity} &{Molecule}
```

(continues on next page)

(continued from previous page)

```
Step_end
res = scfEnX.readProperty(propertyName="SCF_Energy");
res = scfGradX.readProperty(propertyName="Nuclear_Gradient", Property_Base=true);
myGeom.Read();
nAtoms = myGeom.GetNumOfAtoms();

# ----------------------------------------------------------------
# Step 2. Initialize rest of the variables
# ----------------------------------------------------------------
Variable corrGradX[3*nAtoms];   # Correlation part of gradient with basis X
Variable corrGradY[3*nAtoms];   # Correlation part of gradient with basis Y
Variable corrGradCBS[3*nAtoms]; # CBS estimation of correlation part of the␣
↪gradient
Variable scfGradCBS[3*nAtoms];  # CBS estimation of SCF part of the gradient
Variable totalGradCBS[3*nAtoms];# CBS estimation of total gradient

# ---------------------------------------------------
# Step3. Correlation Calculation with small basis set (X)
# ---------------------------------------------------
if (addCorrelation) then
  New_Step
    ! &{method} &{LowerBasis} &{restOfInput}
  Step_end
  res = scfEnX.readProperty(propertyName=scfEnPropName);
  res = corrEnX.readProperty(propertyName=corrEnPropName);
  res = totalGradX.readProperty(propertyName="Nuclear_Gradient", Property_
↪Base=true);

  # ---------------------------------------------------
  # Evaluate correlation gradient with small basis set (X)
  # ---------------------------------------------------
  corrGradX =mat_p_mat(1, totalGradX, -1, scfGradX);
EndIf

# ---------------------------------------------------
# Step4. SCF Calculation with large basis set (Y)
# ---------------------------------------------------
New_Step
  !HF &{UpperBasis} &{restOfInput}
Step_End
res = scfEnY.readProperty(propertyName="SCF_Energy");
res = scfGradY.readProperty(propertyName="Nuclear_Gradient", Property_Base=true);

# ---------------------------------------------------
# Step5. Correlation calculation with large basis set (Y)
# ---------------------------------------------------
if (addCorrelation) then
  New_Step
    ! &{method} &{UpperBasis} &{restOfInput}
  Step_end
  res = scfEnY.readProperty(propertyName=scfEnPropName);
  res = corrEnY.readProperty(propertyName=corrEnPropName);
  res = totalGradY.readProperty(propertyName="Nuclear_Gradient", Property_
↪Base=true);

  # ---------------------------------------------------
  # Evaluate correlation gradient with big basis set Y
  # ---------------------------------------------------
  corrGradY = mat_p_mat(1, totalGradY, -1, scfGradY);
EndIf
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------
# Step6. Extrapolate the SCF part of the gradient and energy
# ------------------------------------------------
eX                = exp(-alpha * sqrt(LowerCardinal));
eY                = exp(-alpha * sqrt(UpperCardinal));
denominator       = eY-eX;

scfEnCBS    = (scfEnX*eY - scfEnY*eX)/(eY-eX);
for i from 0 to scfGradX.GetSize()-1 Do
  gradX = scfGradX[i];
  gradY = scfGradY[i];

  scfGradCBS[i]  = (gradX * eY - gradY * eX)/denominator;
endFor

if (addCorrelation) then
  # ------------------------------------------------
  # Step7. Extrapolate the correlation part of the gradient and energy
  # ------------------------------------------------
  denominator = LowerCardinal^(beta)-(UpperCardinal)^(beta);

  corrEnCBS =  (LowerCardinal^(beta)*corrEnX-(UpperCardinal)^(beta)*corrEnY)/
↪denominator;
  for i from 0 to scfGradX.GetSize()-1 Do
    gradX = corrGradX[i];
    gradY = corrGradY[i];

    corrGradCBS[i] = (LowerCardinal^(beta)*gradX-(UpperCardinal)^(beta)*gradY)/
↪denominator;
  endFor

  # ------------------------------------------------
  # Add SCF and correlation part to get total CBS extrapolated values
  # ------------------------------------------------
  totalGradCBS = mat_p_mat( 1, scfGradCBS, 1, corrGradCBS);
EndIf


# ------------------------------------------------
# Step8. Present the results
# ------------------------------------------------
print( "\n\n\n");
print( "-----------------------------------------------------------\n");
print( "             Compound Extrapolation of Gradient         \n");
print( "-----------------------------------------------------------\n");
print( "Number of atoms       : %d\n", nAtoms);
print( "Lower basis set       : %s\n", LowerBasis);
print( "Upper basis set       : %s\n", UpperBasis);
print( "Alpha                 : %.2lf\n", alpha);
print( "Beta                  : %.2lf\n", beta);
print( "Lower Cardinal number : %d\n", LowerCardinal);
print( "Upper Cardinal number : %d\n", UpperCardinal);
print( "Method                : %s\n", method);
print( "AddCorrelation        : %s\n", AddCorrelation.GetString());
print( "Produce EnGrad File   : %s\n", produceEnGradFile.GetString());
print( "\n\n");
print( "SCF Energy with small basis set         : %.12e\n", scfEnX);
print( "SCF Energy with big basis set           : %.12e\n", scfEnY);
print( "Extrapolated SCF energy                 : %.12e\n", scfEnCBS);
print("\n\n");
if (addCorrelation) then
```

(continues on next page)

(continued from previous page)

```
  print( "Correlation Energy with small basis set : %.12e\n", corrEnX);
  print( "Correlation Energy with big basis set   : %.12e\n", corrEnY);
  print( "Extrapolated correlation  energy        : %.12e\n", corrEnCBS);
  print("\n\n");
  print( "Total Energy with small basis set : %.12e\n", scfEnX   + corrEnX);
  print( "Total Energy with big basis set   : %.12e\n", scfEnY   + corrEnY);
  print( "Extrapolated Total energy         : %.12e\n", scfEnCBS + corrEnCBS);
  print("\n\n");
else
  print( "Total Energy with small basis set : %.12e\n", scfEnX);
  print( "Total Energy with big basis set   : %.12e\n", scfEnY);
  print( "Extrapolated Total energy         : %.12e\n", scfEnCBS);
  print("\n\n");
EndIf

print( "-----------------------------------------------------------------\n");
print( "SCF Gradient with basis set: %s\n", LowerBasis );
print( "-----------------------------------------------------------------\n");
print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
for i from 0 to nAtoms-1 Do
  print("%4d     %20lf     %20lf     %20lf\n", i, scfGradX[3*i], scfGradX[3*i+1],␣
↪scfGradX[3*i+2]);
EndFor
if (addCorrelation) then
  print( "-----------------------------------------------------------------\n");
  print( "Correlation Gradient with basis set: %s\n", LowerBasis );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, corrGradX[3*i],␣
↪corrGradX[3*i+1], corrGradX[3*i+2]);
  EndFor

  print( "-----------------------------------------------------------------\n");
  print( "Total Gradient with basis set: %s\n", LowerBasis );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, totalGradX[3*i],␣
↪totalGradX[3*i+1], totalGradX[3*i+2]);
  EndFor
EndIf

print( "-----------------------------------------------------------------\n");
print( "SCF Gradient with basis set: %s\n", UpperBasis );
print( "-----------------------------------------------------------------\n");
print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
for i from 0 to nAtoms-1 Do
  print("%4d     %20lf     %20lf     %20lf\n", i, scfGradY[3*i], scfGradY[3*i+1],␣
↪scfGradY[3*i+2]);
EndFor

if (addCorrelation) then
  print( "-----------------------------------------------------------------\n");
  print( "Correlation gradient with basis set: %s\n", UpperBasis );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, corrGradY[3*i],␣
↪corrGradY[3*i+1], corrGradY[3*i+2]);
  EndFor
```

(continues on next page)

(continued from previous page)

```
  print( "-----------------------------------------------------------------\n");
  print( "Total Gradient with basis set: %s\n", UpperBasis );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, totalGradY[3*i],␣
↪totalGradY[3*i+1], totalGradY[3*i+2]);
  EndFor
EndIf

print( "-----------------------------------------------------------------\n");
print( "Extrapolated SCF part of the Gradient:\n" );
print( "-----------------------------------------------------------------\n");
print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
for i from 0 to nAtoms-1 Do
  print("%4d     %20lf     %20lf     %20lf\n", i, scfGradCBS[3*i],␣
↪scfGradCBS[3*i+1], scfGradCBS[3*i+2]);
EndFor

if (addCorrelation) then
  print( "-----------------------------------------------------------------\n");
  print( "Correlation Gradient with basis set:\n" );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, corrGradCBS[3*i],␣
↪corrGradCBS[3*i+1], corrGradCBS[3*i+2]);
  EndFor
  print( "-----------------------------------------------------------------\n");
  print( "Total Gradient with basis set:\n" );
  print( "-----------------------------------------------------------------\n");
  print( "Atom     %20s     %20s     %20s\n", "X", "Y", "Z");
  for i from 0 to nAtoms-1 Do
    print("%4d     %20lf     %20lf     %20lf\n", i, totalGradCBS[3*i],␣
↪totalGradCBS[3*i+1], totalGradCBS[3*i+2]);
  EndFor
EndIf
print( "-----------------------------------------------------------------\n");


if (produceEnGradFile) then
  # ----------------------------------------
  # Read the geometry of the last calculation
  # ----------------------------------------
  myGeom.Read();
  Cartesians = myGeom.GetCartesians();
  atomicNumbers = myGeom.GetAtomicNumbers();
  EnGradFile = openFile(enGradFilename, "w");
  Write2File(EnGradFile, "\n\n\n");
  Write2File(EnGradFile, " %d\n", nAtoms);
  Write2File(EnGradFile, "\n\n\n");
  if (addCorrelation) then
    Write2File(EnGradFile, " %.12lf\n", scfEnCBS + corrEnCBS);
  else
    Write2File(EnGradFile, " %.12lf\n", scfEnCBS);
  EndIf
  Write2File(EnGradFile, "\n\n\n");
  for i from 0 to 3*nAtoms-1 Do
    if (addCorrelation) then
      Write2File(EnGradFile, "     %20.12lf\n", totalGradCBS[i]);
    else
```

(continues on next page)

(continued from previous page)

```
      Write2File(EnGradFile, "     %20.12lf\n", scfGradCBS[i]);
    EndIf
  EndFor
  Write2File(EnGradFile, "\n\n\n");
  for i from 0 to nAtoms-1 Do
    Write2File(EnGradFile, "%5d  %12.8lf  %12.8lf   %12.8lf\n", atomicNumbers[i],␣
↪cartesians[i][0], cartesians[i][1], cartesians[i][2]);
  EndFor
  closeFile(EnGradFile);

EndIf

End
```

Comments

8.4.9 BSSE Optimization

Introduction

This script optimizes the geometry of a molecule using gradients corrected for Basis Set Superposition Error (BSSE) correction. The basic step is the usage of a second script that calculates BSSE corrected gradients.

Filename

BSSEOptimization.cmp

SCRIPT

```
# Author:  Frank Neese and Dimitrios G. Liakos
# Date  :  May of 2024
# ---------------------------------------------------
#
# This is a script that will use a compound script to
#   calculate BSSE corrected gradients and use them
#   in combination with ORCA External Optimizer to
#   perform a geometry optimization.
#
# We perform the following steps.
# 1. Choose a compound script that calculates the BSSE
#    corrected gradient.
#    We achieve this with the compoundFilename
#
# 2. Create a script to run an ORCA calculalation with
#    the external optimizer and the BSSE cprrected
#    gradient. We do that by running a script that runs
#    an ORCA calculation that calculates the gradient
#    and then copy this gradient file back to the expected
#    name
#
# 3. Make a normal ORCA New_Step that calls the external
#    optimizer
#
# NOTE: Depending on the chosen method the property names of
#       myPropName has to be adjusted. For the gradient we do
#       not have this problem because we read the last
#       available in the corresponding property file.
#
# NOTE: Variable baseFilename should have the name of the calling
#       orca input file!
#
```

(continues on next page)

(continued from previous page)

```
# -----------------------    Variables    -------------------------------
# --- Variables to be adjusted (e.g. using 'with' -----------------------
Variable molecule           = "01.xyz";         # xyz file of the initial structure
Variable method             = "BP86";            # The method we use for the␣
↪calculation
Variable basis              = " ";               # The basis set
Variable restOfInput        = "";                # The rest of the simple input
Variable charge             = 0;                 # Charge
Variable mult               = 1;                 # Spin multiplicity
Variable myPropName         = "SCF_Energy";      # The name of the property for␣
↪the energy
variable myFilename         = "compoundBSSE";    # Name for the created xyz files
Variable baseFilename       = "run";
Variable gradCreateFile     = "BSSEGradient.cmp";# The compound script that␣
↪extrapolates the gradient
Variable DoOptimization     = false;             # Optimize the monomers or not
Variable produceEnGradFile  = true;              # Produce an ORCA engrad file
Variable enGradFilename     = "result.engrad";   # Filename of the ORCA engrad file
# -------------------------------------------------------
#
#           Variables for the driver script
Variable createDriverScript = true;            # The shell script driver
Variable driverScript;                         # A script to create the␣
↪extrapolated energy gradient
Variable driverScriptName   = "runningScript";
Variable submitCommand      = "orca";
# -------------------------------------------------------

# -------------------------------------------------------
#
#           Variables for the ORCA input
Variable createORCAInput = true;
Variable orcaInput;                           # The ORCA input for the gradient␣
↪extrapolation
Variable orcaInputName  = "runGradient.inp";
# -------------------------------------------------------

# ------------------------------------------------
# 1. Maybe Create the necessary driver script
#    for the external optimizer and make it executable
#    NOTE: This will depenend on the operating system
# ------------------------------------------------
if (createDriverScript) then
  driverScript = openFile(driverScriptName, "w");
  write2File(driverScript, "source ~/.bashrc\n");
  write2File(driverScript, "%s %s\n", submitCommand, orcaInputName );
  write2File(driverScript, "cp %s  %s_Compound_1_EXT.engrad\n", engradFilename,␣
↪baseFilename);
  closeFile(driverScript);
  sys_cmd("chmod +x %s",driverScriptName);
EndIf

# ------------------------------------------------
# 2. Maybe Create the ORCA input that will run the
#    compound script for the gradient extrapolation
# ------------------------------------------------
if (createORCAInput) then
  orcaInput = openFile(orcaInputName, "w");
  Write2File(orcaInput, "%%Compound \"%s\"\n", gradCreateFile);
  Write2File(orcaInput, "  with\n");
  Write2File(orcaInput, "    molecule          =\"%s_Compound_1_EXT.xyz\"\;\n",␣
```

(continues on next page)

(continued from previous page)

```
↪baseFilename);
  Write2File(orcaInput, "    charge            = %d\;\n",     charge);
  Write2File(orcaInput, "    mutliplicity      = %d\;\n",     mult);
  Write2File(orcaInput, "    method            = \"%s\"\;\n", method);
  Write2File(orcaInput, "    basis             = \"%s\"\;\n", basis);
  Write2File(orcaInput, "    restOfInput       = \"%s\"\;\n", restOfInput);
  Write2File(orcaInput, "    myPropName        = \"%s\"\;\n", myPropName);
  Write2File(orcaInput, "    myFilename        = \"%s\"\;\n", myFilename);
  Write2File(orcaInput, "    removeFiles       = false\;\n");
  Write2File(orcaInput, "    DoOptimization    = %s\;\n",     DoOptimization.
↪GetString());
  Write2File(orcaInput, "    produceEnGradFile = %s\;\n",     produceEnGradFile.
↪GetString());
  Write2File(orcaInput, "    enGradFilename    = \"%s\"\;\n", enGradFilename);
  Write2File(orcaInput, "End\n");
  closeFile(orcaInput);
EndIf

# ------------------------------------------------
# 3. Copy the initial XYZ file to the one needed
#    for the external optimizer
# ------------------------------------------------
sys_cmd("cp %s %s_Compound_1_EXT.xyz", molecule, baseFilename);


# ------------------------------------------------
# 1. Run the driver ORCA input file that calls the
#    External optimizer
# ------------------------------------------------
New_Step
  !ExtOpt Opt
  *xyzfile &{charge} &{mult} &{baseFilename}_Compound_1_EXT.xyz
  %method
    ProgExt "./&{driverScriptName}"
  End
Step_End

End
```

Comments

The initial structure should contain some ghost atoms.

8.4.10 Umbrella script

Introduction

This script calculates the potential for the "umbrella effect" in NH_3. In addition it locates the minima and maxima in the potential surface.

Filename

Umbrella.cmp

SCRIPT

```
# ------------------------------------------------
# Umbrella coordinate mapping for NH3
# Author: Frank Neese
# ------------------------------------------------
variable JobName = "NH3-umbrella";
```

(continues on next page)

(continued from previous page)

```
variable amin    = 50.0;
variable amax    = 130.0;
variable nsteps  = 21;
Variable energies[21];

Variable angle;
Variable JobStep;
Variable JobStep_m;
variable step;

Variable method = "BP86";
Variable basis  = "def2-SVP def2/J";

step  = 1.0*(amax-amin)/(nsteps-1);

# Loop over the number of steps
# ----------------------------
for iang from 0 to nsteps-1 do
  angle    = amin + iang*step;
  JobStep  = iang+1;
  JobStep_m= JobStep-1;
  if (iang>0) then
    Read_Geom(JobStep_m);
    New_step
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom constraints
        {A 1 0 2 &{angle} C}
        {A 1 0 3 &{angle} C}
        {A 1 0 4 &{angle} C}
        end
      end

    Step_End
  else
    New_step
      ! &{method} &{basis} TightSCF Opt
      %base "&{JobName}.step&{JobStep}"
      %geom constraints
        {A 1 0 2 &{angle} C}
        {A 1 0 3 &{angle} C}
        {A 1 0 4 &{angle} C}
        end
      end

      * int 0 1
      N 0 0 0 0.0 0.0 0.0
      DA 1 0 0 2.0 0.0 0.0
      H 1 2 0 1.06 &{angle} 0.0
      H 1 2 3 1.06 &{angle} 120.0
      H 1 2 3 1.06 &{angle} 240.0
      *
    Step_End
  endif
   energies[iang].readProperty(propertyName="SCF_ENERGY");
   print(" index: %3d Angle %6.2lf Energy: %16.12lf Eh\n", iang, angle,␣
↪energies[iang]);
EndFor

# Print a summary at the end of the calculation
# ---------------------------------------------
```

(continues on next page)

(continued from previous page)

```
print("//////////////////////////////////////////////////////////\n");
print("// POTENTIAL ENERGY RESULT\n");
print("//////////////////////////////////////////////////////////\n");
variable minimum,maximum;
variable Em,E0,Ep;
variable i0,im,ip;
for iang from 0 to nsteps-1 do
  angle   = amin + 1.0*iang*step;
  JobStep = iang+1;
  minimum = 0;
  maximum = 0;
  i0      = iang;
  im      = iang-1;
  ip      = iang+1;
  E0      = energies[i0];
  Em      = E0;
  Ep      = E0;
  if (iang>0 and iang<nsteps-1) then
    Em = energies[im];
    Ep = energies[ip];
  endif
  if (E0<Em and E0<Ep) then minimum=1; endif
  if (E0>Em and E0>Ep) then maximum=1; endif
  if (minimum = 1 ) then
    print(" %3d  %6.2lf %16.12lf (-)\n",JobStep,angle, E0 );
  endif
  if (maximum = 1 ) then
    print(" %3d  %6.2lf %16.12lf (+)\n",JobStep,angle, E0 );
  endif
  if (minimum=0 and maximum=0) then
    print(" %3d  %6.2lf %16.12lf    \n",JobStep,angle, E0 );
  endif
endfor
print("//////////////////////////////////////////////////////////\n");

End # end of compound block
```

8.4.11 Multi reference

Introduction

This is a script that calculates the atomic electron denstities in free atoms and makes a library of them.

Filename

atomDensities.inp

SCRIPT

```
# FN 07/2024
#
# A compound script to run calculation on free atoms
# in order to generate a library of electron densities
#
%compound
   Variable Element = {" ",
                        "H",                                "He",
                        "Li","Be","B" ,"C" ,"N" ,"O" ,"F" ,"Ne"
                        };
   Variable Nact    = {" ",
                        "1" ,                               "0",
```

(continues on next page)

(continued from previous page)

```
                        "1" ,"0" ,"3" ,"4" ,"5" ,"6" ,"7" ,"0"
                        };

   Variable Norb    = {" ",
                        "1" ,                                "0",
                        "1" ,"0" ,"4" ,"4" ,"4" ,"4" ,"4" ,"0"
                        };

   Variable Nroots  = {" ",
                        "1" ,                                "0",
                        "1" ,"0" ,"3" ,"3" ,"1" ,"3" ,"3" ,"0"
                        };

   Variable Charge  = {" ",
                       "0" ,                              "0",
                       "0" ,"0" ,"0" ,"0" ,"0" ,"0" ,"0" ,"0"
                       };

   Variable Mult    =  {" ",
                        "2" ,                                "1",
                        "2" ,"1" ,"2" ,"3" ,"4" ,"3" ,"2" ,"1"
                        };

   Variable HFTyp   = {" ",
                        "UHF",
↪       "RHF",
                        "CASSCF","RHF","CASSCF" ,"CASSCF" ,"CASSCF" ,"CASSCF" ,
↪"CASSCF" ,"RHF"
                        };

   Variable el;
   for el from 1 to Element.GetSize()-1 do
      if (HFTyp[el]="CASSCF") then
        New_Step
          ! cc-pVDZ VeryTightSCF Conv
          %base "atom_&{Element[el]}_&{Charge[el]}_&{Mult[el]}"
          %casscf  nel    = &{Nact[el]};
                   norb   = &{Norb[el]};
                   nroots = &{Nroots[el]};
                   mult   = &{Mult[el]};
                   end
         * xyz &{Charge[el]} &{Mult[el]}
          &{Element[el]} 0.0  0.0   0.0
         *
        Step_end
      else
        New_Step
         ! &{HFTyp[el]} cc-pVDZ VeryTightSCF Conv
          %base "atom_&{Element[el]}_&{Charge[el]}_&{Mult[el]}"
         * xyz &{Charge[el]} &{Mult[el]}
          &{Element[el]} 0.0  0.0   0.0
         *
        Step_end
      endif
   endfor

endrun;
```

Comments

Here, it's interesting to note that depending on the selected atom, the script either performs a CASSCF calculation, which provides details such as the number of electrons and number of roots, among other parameters, or it carries

out a simple Hartree-Fock calculation.

8.4.12 GoTo

Introduction

This is a brief example demonstrating how the *GoTo* command can be used in *Compound*.

Filename

goTo_Example.inp

SCRIPT

```
# Compound Example on GoTo usage
# Efficient ON/OFF switch

%Compound
  Variable switch="OFF";
  Variable turnOff, turnOn, loopEnd;
  Variable maxIter = 10;
  for i from 0 to maxIter do
    if (switch="ON") then
      GoTo turnOff;
    else
      GoTo turnOn;
    endIf
    turnOff:
      print("Switch: %s\n", switch);
      switch="OFF";
      GoTo loopEnd;
    turnON:
      print("Switch: %s\n", switch);
      switch="ON";
      GoTo loopEnd;
    loopEnd:
  EndFor
End
```

CHAPTER

NINE

UTILITIES AND VISUALIZATION

9.1 Orbital and Density Plots

There are two types of graphics output possible in ORCA - two dimensional contour plots and three dimensional surface plots. The quantities that can be plotted are the atomic orbitals, molecular orbitals, natural orbitals, the total electron density or the total spin density. The graphics is controlled through the block `%plots`.

9.1.1 Contour Plots

The contour plots are controlled via the following variables

```
%plots
  #*** the vectors defining the cut plane
  v1 0, 0, 0  # pointer to the origin
  v2 1, 0, 0  # first direction
  v3 0, 1, 0  # second direction
  #*** alternative to defining vectors. Use atom coordinates
  at1 0 # first atom defining v1
  at2 2 # second atom defining v2
  at3 4 # third atom defining v3
  #*** resolution of the contour
  dim1  45   # resolution in v2-direction
  dim2  45   # resolution in v3-direction
  #*** minimum and maximum values along v2 and v3
  min1  -7.0 # min value along v2 in bohr
  max1   7.0 # min value along v2 in bohr
  min2  -7.0 # min value along v3 in bohr
  max2   7.0 # max value along v3 in bohr
  #***
  UseCol   true   # Use color in the plot (blue=positive,
                  # red=negative)
  Skeleton true   # Draw Skeleton of the molecule of those
                  # atoms that are in or close to the cut
                  # plane
  Atoms    true   # Draw the atoms that are in the plane as
                  # circles
  NCont    200    # Number of contour levels.
  ICont    0      # Draw NCont equally space contours
           1      # Start with 1/NCont and the double the
                  # value for each additional contour
  #*** the format of the output file
  Format   Origin   # straight ascii files
           HPGL     # plotter language files
  #*** the quantities to plot
  MO("MyOrbital-15xy.plt",15,0);  # orbital to plot
  v3= 0, 0, 1                     # change cut plane
  MO("MyOrbital-16xz.plt",16,0);  # orbital to plot
```

(continues on next page)

(continued from previous page)

```
  ElDens("MyElDens.plt");          # Electron density
  SpinDens("MySpinDens");          # Spin density
end
```

The input was:
`v1 = 0, 0, 0; v2 = 1, 0, 0; v3 = 0, 1, 0; min1= -8; max1= 8; min2= -8; max2= 8; dim1= 50; dim2=50; Format = HPGL; NCont = 200; Icont = 1; Skeleton= true; Atoms = true; MO("Test-DFT-H2CO+-MO7xy.plt",7,1);`

NOTE:

- The command `MO("MyOrbital-15xy.plt",15,0);` is to be interpreted as follows: `MO` means that a MO is to be plotted. "`MyOrbital-15xy.plt`" is the file to be created. `15` is the number of the MO to be drawn (remember: counting starts at orbital 0!) and `0` is the operator the orbital belongs to. For a RHF (or RKS) calculation there is only one operator which has number 0. For a UHF (or UKS) calculation there are two operators - the spin-up orbitals belong to operator 0 and the spin-down orbitals belong to operator 1. For ROHF calculations there may be many operators but at the end all orbitals will be collected in one set of vectors. Thus the operator is always =0 in ROHF.
- The `ELDENS` (plot of the total electron density) and `SPINDENS` (plot of the total spin density) commands work analogous to the `MO` with the obvious difference that there is no MO or operator to be defined.

- Analogous to ELDENS and SPINDENS, post-HF densities can be selected using the keyword extended by the respective method. ELDENSMDCI / SPINDENSMDCI will plot the MDCI density, of course only if is available. ELDENSMP2RE and SPINDENSMP2RE will work with the MP2 relaxed density, while ELDENSMP2UR and SPINDENSMP2UR will yield the MP2 unrelaxed density. The OO-RI-MP2 densities can be requested by ELDENSOO or SPINDENSOO. Similarly, AutoCI relaxed densities can be plotted by using the ELDENSAUTOCIRE and SPINDENSAUTOCIRE keywords, and the unrelaxed densities by using ELDENSAUTOCIUR and SPINDENSAUTOCIUR.
- The UNO option plots natural orbitals of the UHF wavefunction (if they are available). No operator can be given for this command because there is only one set of UHF-NOs. Similarly, using UCO option can be used to plot the UHF corresponding orbitals.
- If the program cannot find the plot module ("Bad command or filename") try to use ProgPlot="orca_plot.exe" in the %method block or point to the explicit path.
- The defining vectors v2 and v3 are required to be orthonormal. The program will use a Schmidt orthonormalization of v3 with respect to v2 to ensure orthonormality. If you do not like this make sure that the input vectors are already orthogonal.
- at1, at2 and at3 can be used instead of v1, v2 and v3. In this case say v1 is taken as the coordinates of atom at1. Mixed definitions where say v2 is explicitly given and say v3 is defined through at3 are possible. A value of -1 for at1, at2 and at3 signals that at1, at2 and at3 are not to be used. This type of definition may sometimes be more convenient.
- Variables can be assigned several times. The "actual" value a variable has is stored together with the command to generate a plot (MO, ELDENS or SPINDENS). Thus after each plot command the format or orientation of the plot can be changed for the next one.
- The Origin format produces a straightforward ASCII file with x, y and z values that can be read into your favorite contour plot program or you could write a small program that reads such files and converts them to whatever format is more appropriate for you.
- I usually use Word for Windows to open the HPGL files which appears to work fine. Double clicking on the graphics will allow modification of linewidth etc. For some reason that is not clear to me some graphics programs do not like the HPGL code that is produced by ORCA. If you are an HPGL expert and you have a suggestion - let me know.

9.1.2 Surface Plots

General Points

Surface plots can, for example, be created through an interface to Leif Laaksonen's ***gOpenMol*** program. This program can be obtained free of charge over the internet. It runs on a wide variety of platforms, is easy to use, produces high quality graphics and is easy to interface[1] - thank you Leif for making this program available!

The relevant [PLOTS] section looks like this:

```
%output
  XYZFile  true
end

%plots
  dim1  45   # resolution in x-direction
  dim2  45   # resolution in y-direction
  dim3  45   # resolution in z-direction
  min1  -7.0 # x-min value in bohr
  max1   7.0 # x-min value in bohr
  min2  -7.0 # y-min value in bohr
  max2   7.0 # y-max value in bohr
```

(continues on next page)

[1] There were some reports of problems with the program on Windows platforms. Apparently it is better to choose the display settings as "true color 32 bit" rather than "high 16 bit". Thanks to Thomas Brunold!

(continued from previous page)

```
  min3  -7.0 # z-min value in bohr
  max3   7.0 # z-max value in bohr
  Format  gOpenMol_bin        # binary *.plt file
          gOpenMol_ascii      # ascii *.plt file
          Gaussian_Cube       # Gaussian-cube format
                              # (an ASCII file)
  MO("MyOrbital-15.plt",15,0);  # orbital to plot
  MO("MyOrbital-16.plt",16,0);  # orbital to plot
  UNO("MyUNO-48.plt",48);       # UHF-NO to plot
  ElDens("MyElDens.plt");       # Electron density
  SpinDens("MySpinDens.plt");   # Spin density
end
```

Note

- it is admittedly inconvenient to manually input the dimension of the cube that is used for plotting. If you do nothing such that `min1 = max1 = min2 = max2 = min3 = max3=0` then the program will try to be smart and figure out a good cube size by itself. It will look at the minimum and maximum values of the coordinates and then add 7 bohrs to each dimension in the hope to properly catch all wavefunction tails.

Sometimes you will want to produce orbital plots after you looked at the output file and decided which orbitals you are interested in. In this case you can also run the `orca_plot` program in a crude interactive form by invoking it as:

```
orca_plot  MyGBWFile.gbw  -i
```

This will provide you with a subset of the capabilities of this program but may already be enough to produce the plots you want to look at. Note that for the name of the GBW-file you may as well input files that result from natural orbitals (normally `*.uno`), corresponding orbitals (normally `*.uco`) or localized orbitals (normally `*.loc`). Once in the interactive program, by entering '1' for 'Enter type of plot,' you will access a list of available plot capabilities relevant to your current calculation file (MyGBWFile.gbw):

```
-----------------------------------------------------------------------
Plot-Type is presently: 1
-----------------------------------------------------------------------
Searching for Ground State Electron or Spin Densities:              ...
-----------------------------------------------------------------------
     1 -   molecular orbitals
     2 -   (scf) electron density                              ......  (scfp      ␣
↪              )  => AVAILABLE
     3 -   (scf) spin density                                  ......  (scfr      ␣
↪              )  => AVAILABLE
     4 -   natural orbitals
     5 -   corresponding orbitals
     6 -   atomic orbitals
     7 -   mdci electron density                               ......  (mdcip     ␣
↪              )  - NOT AVAILABLE
     8 -   mdci spin density                                   ......  (mdcir     ␣
↪              )  - NOT AVAILABLE
     9 -   OO-RI-MP2 density                                   ......  (pmp2re    ␣
↪              )  - NOT AVAILABLE
    10 -   OO-RI-MP2 spin density                              ......  (pmp2ur    ␣
↪              )  - NOT AVAILABLE
    11 -   MP2 relaxed density                                 ......  (pmp2re    ␣
↪              )  - NOT AVAILABLE
    12 -   MP2 unrelaxed density                               ......  (pmp2ur    ␣
↪              )  - NOT AVAILABLE
```

(continues on next page)

(continued from previous page)

```
    13 -   MP2 relaxed spin density                          ......  (rmp2re    ␣
↪              )  - NOT AVAILABLE
    14 -   MP2 unrelaxed spin density                        ......  (rmp2ur    ␣
↪              )  - NOT AVAILABLE
    15 -   LED dispersion interaction density                ......  (ded21     ␣
↪              )  - NOT AVAILABLE
    16 -   Atom pair density
    17 -   Shielding Tensors
    18 -   Polarisability Tensor
    19 -   AutoCI relaxed density                            ......  (autocipre ␣
↪              )  - NOT AVAILABLE
    20 -   AutoCI unrelaxed density                          ......  (autocipur ␣
↪              )  - NOT AVAILABLE
    21 -   AutoCI relaxed spin density                       ......  (autocirre ␣
↪              )  - NOT AVAILABLE
    22 -   AutoCI unrelaxed spin density                     ......  (autocirur ␣
↪              )  - NOT AVAILABLE
-----------------------------------------------------------------------
Searching for State or Transition State AO Electron Densities:      ...
-----------------------------------------------------------------------
    23 -   CIS unrelaxed transition AO density               ......  (Tdens-CIS ␣
↪              )  - NOT AVAILABLE
    24 -   ROCIS unrelaxed transition AO density             ......  (Tdens-
↪ROCIS              )  - NOT AVAILABLE
    25 -   CAS unrelaxed transition AO density               ......  (Tdens-CAS ␣
↪              )  - NOT AVAILABLE
    26 -   ICE unrelaxed transition AO density               ......  (Tdens-ICE ␣
↪              )  - NOT AVAILABLE
    27 -   MRCI unrelaxed transition AO density              ......  (Tdens-MRCI␣
↪              )  - NOT AVAILABLE
    28 -   LFT unrelaxed transition AO density               ......  (Tdens-LFT ␣
↪              )  - NOT AVAILABLE
-----------------------------------------------------------------------
Searching for State or Transition State MO Electron Densities:      ...
-----------------------------------------------------------------------
    29 -   CIS unrelaxed transition MO density               ......  (Tdens-
↪CISMO              )  - NOT AVAILABLE
    30 -   ROCIS unrelaxed transition MO density             ......  (Tdens-
↪ROCISMO            )  - NOT AVAILABLE
    31 -   CAS unrelaxed transition MO density               ......  (Tdens-
↪CASMO              )  - NOT AVAILABLE
    32 -   ICE unrelaxed transition MO density               ......  (Tdens-
↪ICEMO              )  - NOT AVAILABLE
    33 -   MRCI unrelaxed transition MO density              ......  (Tdens-
↪MRCIMO             )  - NOT AVAILABLE
    34 -   LFT unrelaxed transition MO density               ......  (Tdens-
↪LFTMO              )  - NOT AVAILABLE
-----------------------------------------------------------------------
Searching for State or Transition State QDPT AO Electron Densities: ...
-----------------------------------------------------------------------
    35 -   CAS QDPT unrelaxed transition AO density          ......  (Tdens-
↪CASQDSOC           )  - NOT AVAILABLE
    36 -   DCDCAS QDPT unrelaxed transition AO density       ......  (Tdens-
↪CASDCDQDSOC        )  - NOT AVAILABLE
    37 -   CAS CUSTOM E QDPT unrelaxed transition AO density ......  (Tdens-
↪CASCUSTOMEQDSOC    )  - NOT AVAILABLE
    38 -   NEVPT2 QDPT unrelaxed transition AO density       ......  (Tdens-
↪CASPTQDSOC         )  - NOT AVAILABLE
    39 -   QDNEVPT2 QDPT unrelaxed transition AO density     ......  (Tdens-
↪CASQDPTQDSOC       )  - NOT AVAILABLE
    40 -   MRCI QDPT unrelaxed transition AO density         ......  (Tdens-
```

(continues on next page)

(continued from previous page)

```
↪MRCIQDSOC          )  - NOT AVAILABLE
   41 -   ROCIS QDPT unrelaxed transition AO density          ......  (Tdens-
↪ROCISQDSOC         )  - NOT AVAILABLE
   42 -   LFT QDPT unrelaxed transition AO density            ......  (Tdens-
↪LFTQDSOC           )  - NOT AVAILABLE
```

Fig. 9.1: The π^* orbital of H_2CO as calculated by the RI-BP/VDZP method.

See also

The visualization of fractional occupation weighted densities (FOD) is described in Section 5.3.2.

Interface to gOpenMol

Here is a short summary of how to produce these plots with gOpenMol:

- First of all the molecular geometry must be save by choosing `XYZFile=true` in the `[OUTPUT]` block. This will produce a straightforward ascii file containing the molecular geometry (or simply `! XYZFile`).
- After having produced the plot files start ***gOpenMol*** and choose ***File-Import-Coords*** . In the dialog choose the *XYZ* format and select the file. Then press ***apply*** and ***dismiss*** . The molecule should now be displayed in the graphics window.
- You can change the appearance by choosing ***View-Atom type*** .
- The color of the background can be changed with ***Colour-Background*** .
- After having done all this choose ***Plot-Contour*** and select the ***Browse*** button to select the appropriate file. Then press ***Import File*** to read it in. NOTE: you can only directly read files that were produced in `gOpenMol_bin` format. If you have chosen `gOpenMol_ascii` you must first use the ***gOpenMol*** file conversion utility under ***Run-Pltfile (conversion)*** to produce the binary plt file.
- After having read the plt file choose the appropriate isocontour value (one for the positive and one for the negative lobes of an orbital) and select suitable colors via ***Colour(n)*** to the right of the isocontour value. The ***Details*** button allows you to choose between solid and mesh representation and other things.
- Once the plot looks the way you like, use ***File-Hardcopy*** to produce a publication quality postscript or bitmap picture that can be imported into any word processing or graphics software.

Interface to Molekel

The Molekel program (http://ugovaretto.github.io/molekel/) is another beautiful and easy-to-use graphics tool that is recommended in combination with ORCA. You may even find it a little easier to use than gOpenMol but this may be a matter of personal taste. In order to produce plots with Molekel follow the following procedure:

- Produce Gaussian-Cube files (and optionally also an XYZ file) with ORCA as described above.
- Start Molekel and use the right mouse button to obtain the **Load** menu.
- Choose the format **xyz** to load your coordinates
- Use the right mouse button again to select the **Surface** menu
- Choose the format "Gaussian Cube" and click **Load Surface**
- Click on **Both Signs** if you visualize an orbital or spin density
- Select a suitable contour value in the **Cutoff** box.
- Click on **Create Surface**. That's it!
- In the **Color** menu (also available via the right mouse button) you can adjust the colors and in the main menu the display options for your molecule. Default settings are in a startup file that you can modify to suit your taste. More details are in the Molekel manual – check it out; it can do many other useful things for you too!

Visualization of three-dimensional representation of MOs, natural orbitals, electron densities, and spin densities is usually more intuitive than examining MO coefficients and it is is described in detail in section *Orbital and Density Plots*. The files necessary for such visualizations can be readily generated with ORCA in various ways and then opened in visualization software such as `gOpenMol` and `Molekel`.[2] In the following example, we briefly describe visualization of MOs.

To visualize MOs with`gOpenMol`, the `plt` file of MOs can be generated in the `gOpenMol_bin` format from the `gbw` file using `orca_plot` utility program or directly from the ORCA run through the `%plots` block of the input file:

[2] The `Molekel` developers ask for the following citation – please do as they ask:
MOLEKEL 4.2, P. Flukiger, H.P. Lüthi, S. Portmann, J. Weber, Swiss Center for Scientific Computing, Manno (Switzerland), 2000-2006.
S. Portmann, H.P.Łüthi. MOLEKEL: An Interactive Molecular Graphics Tool. CHIMIA (2000), 54, 766-770. The program appears to be maintained by Ugo Varetto at this time.

```
! HF def2-SVP XYZFile

%plots Format gOpenMol_bin
       MO("CO-4.plt",4,0);
       MO("CO-8.plt",8,0);
       end

*xyz 0 1
 C  0  0  0
 O  0  0  1.13
*
```

In this input file, the `MO("CO-4.plt",4,0);` command is used to evaluare MO labeled as 4 for operator 0 and then to strore it in the "CO-4.plt" file. For RHF and ROHF, one should always use operator 0. For UHF, operators 0 and 1 correspond to spin-up and spin-down orbitals, respectively.

When the produced `plt` files are opened with `gOpenMol` (see section *Surface Plots* for details), the textbook-like π and π^* MOs of the CO molecule are visualized as in Figure Fig. 9.2.

(a) (a) (b) (b)

Fig. 9.2: (a) π and (b) π^* MOs of the CO molecule obtained from the interface of `ORCA` to `gOpenMol`.

If the `gOpenMol_ascii` file format was requested, gOpenMol conversion utility or some other tools might then be needed to convert this human-readable file to the machine-readable `gOpenMol_bin` format.

In order to use the interface to `Molekel`, an ASCII file in the `Cube` or `Gaussian_Cube` format needs to be generated. Such ASCII files can be actually transferred between platforms. The `Cube` format can be requested in the `%plots` block as:

```
! HF def2-SVP XYZFile

%plots Format Cube
       MO("CO-4.cube",4,0);
       MO("CO-8.cube",8,0);
       end

*xyz 0 1
 C  0  0  0
```

(continues on next page)

(continued from previous page)

```
 O  0  0  1.13
*
```

To visualize MOs strored in the `*.cube` file, start `Molekel` and, via a right mouse click, load the `*.xyz` file and/or the `*.cube` file. lternatively, navigate to the surface menu, select the "gaussian-cube" format, and load the surface. For orbitals, click the "both signs" button and select a countour value in the "cutoff" field. Then, click "create surface". The colour schemes and other fine details of the plots can be easily adjusted as desired. Finally, create files via the "snapshot" feature of `Molekel`. Figure Fig. 9.3 demonstrates a `Molekel` variant of Figure Fig. 9.2.

(a) (a) (b) (b)

Fig. 9.3: (a) π and (b) π^* MOs of the CO molecule obtained from the interface of `ORCA` to `Molekel`.

It is worth noting that there are several other freeware programs, such as `UCSF CHIMERA`, that can read `Gaussian_Cube` files and provide high-quality plots.

In some situations, visualization of the electronic structure in terms of localized molecular orbitals might be quite helpful. As unitary transformations among occupied orbitals do not change the total wavefunction, such transformations can be applied to the canonical SCF orbitals with no change of the physical content of the SCF wavefunction. The localized orbitals correspond more closely to the pictures of orbitals that chemists often enjoy to think about. Localized orbitals according to the Pipek-Mezey population-localization scheme are quite easy to compute. For example, the following run reproduces the calculations reported by Pipek and Mezey in their original paper for the N_2O_4 molecule.

```
! HF STO-3G Bohrs

%loc
     LocMet  PipekMezey # localization method. Choices:
                          # PipekMezey (=PM)
                          # FosterBoys (=FB)
     T_Core  -1000      # cutoff for core orbitals
     Tol     1e-8       # conv. Tolerance (default=1e-6)
     MaxIter 20         # max. no of iterations (def. 128)
     end

*xyz 0 1
 N    0.000000   -1.653532   0.000000
 N    0.000000    1.653532   0.000000
 O   -2.050381   -2.530377   0.000000
```

(continues on next page)

(continued from previous page)

```
 O    2.050381   -2.530377   0.000000
 O   -2.050381    2.530377   0.000000
 O    2.050381    2.530377   0.000000
*
```

Based on the output file of this job, localized MOs consist of six core like orbitals (one for each N and one for each O), two distinct lone pairs on each oxygen, a σ- and a π-bonding orbital for each N-O bond and one N-N σ-bonding orbital which corresponds to the dominant resonance structure of this molecule. You will also find a file with the extension `.loc` in the directory where you run the calculation. Like the standard `gbw` file, it can used to extract files for plotting or as input for another calculation (warning! The localized orbitals have no well defined orbital energy. If you do use them as input for another calculation use `GuessMode=CMatrix` in the `%scf` block).

9.2 Utility Programs

orca_2aim	Produces WFN and WFX files suitable for AIM analysis
orca_2json	Converts information from the gbw-file into JSON files
orca_2mkl	Produces an ASCII file to be read by molekel, molden or other visualization programs
orca_asa	Calculates band shapes of absorption, fluorescence and resonance Raman spectra
orca_chelpg	Calculates electrostatic potential derived charges
orca_euler	Calculates Euler angles from `.property.txt` file
orca_exportbasis	Prints out any basis set in ORCA or GAMESS-US format
orca_fitpes	Fits potential energy curves of diatomics
orca_mapspc	Produces files for transfer into plotting programs
orca_mergefrag	Merges MO coefficients from two independent .gbw files
orca_pltvib	Produces files for the animation of vibrations
orca_pnmr	Calculates paramagnetic NMR shielding tensors
orca_vib	Calculates vibrational frequencies from a completed frequency run (also used for isotope shift calculations)
otool_gcp	Calculates Geometrical Counterpoise Correction (GCP)
otool_xtb	Allows using standalone xTB code from the Grimme lab with ORCA

Friends of ORCA:

gennbo	The NBO analysis package of Weinhold. It must be purchased separately from the University of Wisconsin. Older versions available for free on the internet may also work.
Molekel	Molecular visualization program (see *Interface to Molekel*)
gOpenMol	Molecular visualization program (see *Interface to gOpenMol*)
Avogadro	Molecular builder and visualization program with ORCA support (login into ORCA Forum and then navigate to Downloads. See also original repository)
Chemcraft	Molecular builder and visualization program with ORCA support (see download page)
SEQCROW	Molecular builder and visualization program with ORCA support (see repository)

Each module is capable of standalone execution; however, it is typically more convenient to perform all operations through the main ORCA driver, which handles the workflow automatically.

ORCA is distributed as a set of precompiled executables, so no formal installation is required. You can simply place the files in a directory of your choice. To ensure that the components can interact correctly, the directory should be added to your system's PATH environment variable. Alternatively, the locations of individual components can be specified manually in the input file.

The xtb tool (recommended version 6.7.1 or higher) must be downloaded separately from the Grimme lab's repository. Once downloaded, the xtb executable should be placed in the same directory as the ORCA binaries.

We make extensive use of Chemcraft for visualization and analysis. Although it is a Windows-based program, it runs well on macOS and Linux systems using Wine a virtual machine, or remote desktop.

Another popular visualization program is ChimeraX together with the SEQCROW plugin.

OpenBabel is a versatile and widely used tool for converting between various chemical file formats.

Finally, Avogadro is an excellent tool to edit molecular geometries. It is also able to generate ORCA input files. The Avogadro version with the latest ORCA modifications is available on the ORCA download site.

Finally, Avogadro is an excellent tool for editing molecular geometries. It can also generate ORCA input files directly. A version of Avogadro that includes the latest ORCA-specific modifications is available on the ORCA Downloads page, which can be accessed after logging into theORCA Forum.

For more information and suggestions, please visit the ORCA Forum.

9.2.1 orca_mapspc

The utility program `orca_mapspc` prepares input files for standard graphics software to plot both the calculated line spectrum and the corresponding convoluted continuous band shape. For examples of plotted spectra generated using the files provided by `orca_mapspc`, see the following sections: *Semiempirical Methods*, *IR Spectra*, *Raman Spectra* and *Resonant Inelastic Scattering Spectroscopy*):

The symbolic command structure of `orca_mapspc` is as follows:

```
orca_mapspc  Output-file Spectrum Options
```

where

```
  Outfile-file     = name of an ORCA output file
                   name of an ORCA Hessian file (for IR and Raman)

  Spectrum         = abs     - Absorption spectra
                     cd      - CD spectra
                     ir      - IR spectra
                     raman   - Raman spectra

  Options          -x0value: Start of the x-axis for the plot
                   -x1value: End of the x-axis for the plot
                   -wvalue : Full-width at half-maximum height in
                             cm**-1 for each transition
                   -nvalue : Number of points to be used
```

For a complete list of supported spectrum types and advanced options, run `orca_mapspc` without arguments in a terminal.

```
orca_mapspc
```

This will display all supported spectrum types and options:

```
--------------------------------------------------------------------------------
↪-----------------------------------------------------------------------
usage: orca_mapspc Output-file {ABS, ABSV, ABSQ, ABSFFMIO, CD, IR, VCD, RAMAN,␣
↪NRVS, VDOS, MCD, CDV, CDFFMIO, SOCCD, SOCCDV,SOCCDFFMIO, SOCABS, SOCABSV,␣
↪SOCABSQ,
                                SOCABSFFMIO, XES, XESV, XESQ, XESFFMIO, XAS, XASV,␣
↪XASQ, XASFFMIO, XESSOC, XESSOCV, XESSOCQ, XESSOCFFMIO,
                                XASSOC, XASSOCV, XASSOCQ, XASSOCFFMIO, RIXS,␣
↪RIXSSOC, TRANSABS, TRANSCD, TRANSSOCABS} -options
--------------------------------------------------------------------------------
↪-----------------------------------------------------------------------
----General options ----
```

(continues on next page)

(continued from previous page)

```
  -o  output file
  -cm use cm**-1 (default)
  -eV use eV (default cm**-1)
  -g0  use Gaussian lineshape default
  -l0  use Lorentz  lineshape (only for ABS, ABSV, ABSQ, ABSFFMIO, SOCABS, SOCABSV,
↪ SOCABSQ, SOCABSFFMIO,  XES, XESV, XESQ,XESFFMIO, XAS, XASV, XASQ, XASFFMIO,␣
↪XESSOC, XASSOC, RIXS, RIXSSOC or Transient spectra)
  -v0  use Voigt lineshape (only for ABS, ABSV, ABSQ, ABSFFMIO, SOCABS, SOCABSV,␣
↪SOCABSQ, SOCABSFFMIO, XES, XESV, XESQ,XESFFMIO, XAS, XASV, XASQ, XASFFMIO,␣
↪XESSOC, XASSOC, RIXS, RIXSSOC, or Transient spectra)
  -x0 initial point of spectrum
  -x1 final   point of spectrum
  -w  line width for Gaussian/Lorenzian linewidth
  -q  line width for the Gaussian part of Voigt linewidth
  -kw coeffitient for the line width calculated as kw*sqrt(energy)
  -n  number of points
----The following additional options are for IR/VCD/RAMAN spectra types----
  -fac   number for shifting the frequencies
----The following additional options are for RIXS and RIXSSOC spectra types----
  -x2 initial point of the spectrum along y axis
  -x3 final   point of the spectrum along yaxis
  -g1, -l1, -v1  line width for Gaussian/Lorenzian linewidth along y axis
  -m  number of points for the emission spectrum
  -eaxis plot option for the emission axis: (1) for Energy transfer (2) for␣
↪emission spectrum
  -uex  number of user defined cuts at constant Excitation Energy axis
  -udw  number of user defined cuts at constant Emission/ Energy Transfer axis
  -dx   number for shifting the spectra along the Excitation /Emission Energy axis
  -kg coeffitient for the line width calculated as kg*sqrt(energy)
  -umem control the memory demand
----Using external files----
  paras.inp: a list of energy ranges with desired broadening parameters
             for x axis: E_start E_stop Width
             for y axis: 0 0 0 E_start E_stop Width
             for xy axis: E_start1 E_stop1 Width1 E_start2 E_stop2 Width2
  udex.inp: a list of energies for taking cuts at constant Excitation Energy axis␣
↪(Only for RIXS/RIXSOC)
  udem.inp: a list of energies for taking cuts at constant Emission/ Energy␣
↪Transfer axis (Only for RIXS/RIXSOC)
  gfsp.inp: a list of ground-final state pairs to generate individual state pair␣
↪RIXS planes and respective analysis planes (Only for ROCIS RIXS/RIXSOC)
-------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
```

This document provides a detailed explanation of the spectroscopic keywords supported by `orca_mapspc`, grouped by category.

Optical/X-ray Absorption Spectra

These keywords compute absorption spectra for optical or X-ray regions, describing how a molecule absorbs light at specific energies.

Dipole Approximation

The dipole approximation assumes that the interaction is dominated by the electric dipole term.

- **ABS** (Absorption Optical/X-ray Spectra in Dipole Length Representation)
  - *Description*: Plots absorption spectra using the electric dipole operator in length representation $(\mu = -er)$.
  - *Purpose*: Suitable for UV-Vis or X-ray absorption spectroscopy (XAS) where dipole transitions dominate.
  - *Use Case*: Common for any chemical system in the UV-visible range or XAS.
  - *Output*: Intensity (oscillator strength, or Absorbance ϵ for UV/Vis spectra) vs. energy (e.g. wavelength)
  - *Notes*: More accurate for low-energy transitions; less reliable for high-energy X-ray transitions.
  - *Convensions*: Oscilator strength $f_{osc} = 4.31 * 10^{-9} * \epsilon$
- **ABSV** (Absorption Optical Spectra in Dipole Velocity Representation)
  - *Description*: Uses the dipole operator in velocity representation $(\mu \propto p)$.
  - *Purpose*: Alternative to `ABS` for numerical stability or cross-checking.
  - *Use Case*: Validation of `ABS` results.
  - *Output*: Similar to `ABS`.
  - *Notes*: Equivalent to `ABS` for exact wavefunctions but may differ with approximate methods.

Beyond Dipole Approximation

These methods include higher-order interactions (e.g., quadrupole, magnetic dipole) for improved accuracy.

- **ABSFFMIO** (Absorption Spectra with Semiclassical Full Matter Interaction Operator)
  - *Description*: Plots spectra on the basis of the semiclassical full matter interaction operator including terms beyond dipole approximation.
  - *Purpose*: Accurate for systems with significant quadrupole or magnetic dipole contributions
  - *Use Case*: Recommended for routine X-ray absorption spectra (e.g., K-edge XAS …) or optical spectra bearing electric dipole forbidden transitions
  - *Output*: Conventional spectrum table with dipole and beyond contributions. Intensity (oscillator strength, or Absorbance ϵ for UV/Vis spectra) vs. energy (e.g. wavelength)
  - *Notes*: Recomended and robust for most applications.
- **ABSQ** (Absorption Spectra with Dipole, Quadrupole, and Magnetic Dipole)
  - *Description*: Includes electric dipole (length), electric quadrupole, and magnetic dipole contributions multipolar formulations beyond dipole.
  - *Purpose*: Detailed spectra for expert users, especially for forbidden transitions.
  - *Use Case*:X-ray absorption (e.g., transition metal K-edges) where one needs to do complex/advanced analysis.
  - *Output*: Process all available tables including:
    1. **Origin-dependent tables**

2. **Origin-adjusted tables** [670].
3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].

– *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.

Optical/X-ray Circularly Polarized Spectra

These keywords process spectra for circularly polarized light, relevant for circular dichroism (CD) and magnetic circular dichroism ((X)MCD).

- **CD** (Circular Dichroism)
  - *Description*: Plots differential absorption of left- and right-circularly polarized light.
  - *Purpose*: Probes chiral molecules or asymmetric transitions.
  - *Use Case*: UV-Vis spectroscopy for enantiomers or chiral complexes.
  - *Output*: $(\Delta\epsilon)$ vs. energy (e.g. wavelength)
  - *Notes*: Requires a chiral system.
  - *Convensions*: Rotatory strength: $R = 22.94 * \Delta\epsilon$
- **MCD** (Magnetic Circular Dichroism)
  - *Description*: Plots the computed differential of left- and right-circularly polarized light in the presence of magnetic field.
  - *Purpose*: Probes spin-orbit coupling or Zeeman effects in the MCD process
  - *Use Case*: Optical (MCD) or X-ray spectroscopic (XMCD) properies of transition metal complexes or magnetic materials.
  - *Output*: Left-Right (X)MCD intensity vs. energy (e.g. wavelength) and Left+Right ABS/XAS intesity vs. energy (e.g. wavelength)
  - *Notes*: Requires magnetic field simulation runs through DoMCD keyword accross the elligible modules (TD-DFT, CASSCF, MRCI, RASCI, ROCIS, LFT…)
- **CDV** (Circular Dichroism in Velocity Representation)
  - *Description*: `CD` in velocity representation.
  - *Purpose*: Alternative formulation for numerical stability.
  - *Use Case*: Validation of `CD` results.
  - *Output*: Similar to `CD`.
- **CDFFMIO** (Circular Dichroism with Full Matter Interaction Operator)
  - *Description*: Plots `CD` spectra generated employing the semiclassical full matter interaction operator including terms beyond dipole approximation..
  - *Purpose*: Improved accuracy beyond dipole approximation, for CD spectra of nimianally dipole forbidden transitions
  - *Use Case*: Suitable for CD spectroscopy dominated by electric dipole forbidden transitions.
  - *Output*: CD spectrum beyond dipole approximation, $(\Delta\epsilon)$ vs. energy (e.g. wavelength)

Spin-Orbit Coupling (SOC) Variants

These keywords include spin-orbit coupling

1. for chemical systems with significant multiplet structure, (L-edge XAS)
2. for chemical systems bearing heavy elements (Lanthanides, Actinides)
3. in order to probe spin-forbidden transitions.

- **SOCABS** (SOC Absorption Spectra)
  - *Description*: Absorption spectra with SOC in dipole length representation.
  - *Purpose*: Accounts for SOC effects in optical/X-ray absorption.
  - *Use Case*: ABS/XAS spectra where SOC effects dominate.
  - *Output*: Absorption spectrum with SOC effects. Intensity (oscillator strength, or Absorbance ϵ for UV/Vis spectra) vs. energy (e.g. wavelength)
- **SOCABSV** (SOC Absorption Spectra in Velocity Representation)
  - *Description*: `SOCABS` in velocity representation.
  - *Purpose*: Alternative formulation for `SOCABS` for numerical stability or cross-checking
  - *Use Case*: Validation of `SOCABS` results.
  - *Output*: Similar to `SOCABS`.
- **SOCABSFFMIO** (SOC Absorption Spectra with Full Matter Interaction Operator)
  - *Description*: `SOCABS` employing beyond dipole approximation intensity contributions.
  - *Purpose*: Recommended for routine SOC-ABS/XAS beyond dipole approximation.
  - *Use Case*: e.g. XAS for high symmetry TM or heavy element elements chemical systems
  - *Output*: Conventional ABS spectrum beyond dipole approximation of FFMIO operator including SOC corrections. Intensity (oscillator strength, or Absorbance ϵ for UV/Vis spectra) vs. energy (e.g. wavelength)
- **SOCABSQ** (SOC Absorption Spectra with Dipole, Quadrupole, and Magnetic Dipole contributions)
  - *Description*: `SOCABS` beyond dipole approximation intensity contributions.
  - *Purpose*: Detailed tables of SOC absorption spectra in multipolar formulation.
  - *Use Case*: X-ray absorption with SOC (e.g., L-edges) where one needs to do complex/advanced analysis .
  - *Output*: Process all avalable tables including:
    1. **Origin-dependent tables**
    2. **Origin-adjusted tables** [670].
    3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].
  - *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.
- **SOCCD** (SOC Circular Dichroism)
  - *Description*: `CD` with spin-orbit coupling.
  - *Purpose*: Captures SOC in chiral systems.
  - *Use Case*: Optical CD for chemical systems with comple multiple structure, e.g. bering Lanthanide, Actinide metal centers.
  - *Output*: CD spectrum with SOC effects. $(\Delta\epsilon)$ vs. energy (e.g. wavelength)

- **SOCCDV** (SOC Circular Dichroism in Velocity Representation)
  - *Description*: `SOCCD` in velocity representation.
  - *Purpose*: Alternative formulation for `SOCCD`.
  - *Use Case*: Validation or numerical stability.
  - *Output*: Similar to `SOCCD`.
- **SOCCDFFMIO** (SOC Circular Dichroism with Full Matter Interaction Operator)
  - *Description*: `SOCCD` employing beyond dipole approximation intensity contributions.
  - *Purpose*: Recommended for processing routine SOC-CD spectra beyond dipole approximation.
  - *Use Case*: CD studies of high symmetry TM or heavy element elements chemical systems, with complex multiple structure
  - *Output*: CD spectrum beyond dipole approximation of FFMIO operator including SOC corrections. $(\Delta\epsilon)$ vs. energy (e.g. wavelength)

X-ray Emission Spectroscopy (XES)

These keywords plot X-ray emission spectra, generated within the %xes module or correlated methods that can generate XES spectra within the OPE framework.

- **XES** (X-ray Emission Spectra)
  - *Description*: Processing XES in dipole length representation.
  - *Purpose*: Processing spectra generated on the basis X-ray electron decay upon core electron ionization.
  - *Use Case*: X-ray spectroscopy of materials.
  - *Output*: Emission spectrum, Intensity (oscillator strenth) vs. energy (e.g. electron volt)
- **XESV** (X-ray Emission Spectra in Velocity Representation)
  - *Description*: `XES` in velocity representation.
  - *Purpose*: Alternative formulation for `XES`.
  - *Output*: Similar to `XES`.
- **XESFFMIO** (X-ray Emission Spectra with Full Matter Interaction Operator)
  - *Description*: `XES` employing beyond dipole approximation intensity contributions.
  - *Purpose*: Recommended for processing routine XES spectra beyond dipole approximation.
  - *Output*: XES spectrum beyond dipole approximation of FFMIO operator. Intensity (oscillator strenth) vs. energy (e.g. electron volt)
- **XESQ** (X-ray Emission Spectra with Dipole, Quadrupole, and Magnetic Dipole)
  - *Description*: `XES` Including electric dipole (length), electric quadrupole, and magnetic dipole contributions multipolar formulations beyond dipole.
  - *Purpose*: Detailed XES spectra for expert users, where one needs to do complex/advanced analysis.
  - *Output*: As in `ABSQ` process all avalable tables including:
    1. **Origin-dependent tables**
    2. **Origin-adjusted tables** [670].
    3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].
  - *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.

- **XESSOC** (SOC X-ray Emission Spectra)
  - *Description*: `XES` with spin-orbit coupling.
  - *Purpose*: Processing spectra generated on the basis of X-ray electron decay upon core electron ioniztion including SOC effects.
  - *Use Case*: Recommended for processing routine SOC-XES spectra.
  - *Output*: Emission spectrum with SOC effects. Intensity (oscillator strenth) vs. energy (e.g. electron volt)
- **XESSOCV** (SOC X-ray Emission Spectra in Velocity Representation)
  - *Description*: `XESSOC` in velocity representation.
  - *Purpose*: Alternative formulation for `XESSOC`.
  - *Output*: Similar to `XESSOC`.
- **XESSOCFFMIO** (SOC X-ray Emission Spectra with Full Matter Interaction Operator)
  - *Description*: `XESSOC` employing beyond dipole approximation intensity contributions.
  - *Purpose*: Recommended for processing routine SOC-XESbeyond dipole approximation.
  - *Output*: X-ray emission spectrum beyond dipole approximation of FFMIO operator including SOC corrections plotted in terms of Intensity (oscillator strenth) vs. energy (e.g. electron volt)
- **XESSOCQ** (SOC X-ray Emission Spectra with Dipole, Quadrupole, and Magnetic Dipole)
  - *Description*: `XESSOC` with non-dipole contributions.
  - *Purpose*: Detailed SOC-XES spectra where one needs to do complex/advanced analysis.
  - *Output*: As in `SOCABSQ` process all avalable tables including:
    1. **Origin-dependent tables**
    2. **Origin-adjusted tables** [670].
    3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].
  - *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.

X-ray Absorption Spectroscopy (XAS)

These keywords plot X-ray absorption spectra, generated within the %xes module.

- **XAS** (X-ray Absorption Spectra)
  - *Description*: Plots XAS in dipole length representation.
  - *Purpose*: Processing spectra probing core electron excitations (e.g., K-edge).
  - *Use Case*: 1 electron DFT X-ray Absoprtion spectra.
  - *Output*: Absorption spectrum for X-ray energies. (Intensity (oscillator strenth) vs. energy (e.g. electron volt))
- **XASV** (X-ray Absorption Spectra in Velocity Representation)
  - *Description*: `XAS` in velocity representation.
  - *Purpose*: Alternative formulation for `XAS`.
  - *Output*: Similar to `XAS`.
- **XASFFMIO** (X-ray Absorption Spectra with Full Matter Interaction Operator)
  - *Description*: `XAS` employing beyond dipole approximation intensity contributions.

  - *Purpose*: Recommended for processing routine XAS spectra beyond dipole approximation.
  - *Output*: XAS spectrum beyond dipole approximation of FFMIO operator (Intensity (oscillator strenth) vs. energy (e.g. electron volt)).

- **XASQ** (X-ray Absorption Spectra with Dipole, Quadrupole, and Magnetic Dipole)
  - *Description*: `XASS` Including electric dipole (length), electric quadrupole, and magnetic dipole contributions multipolar formulations beyond dipole.
  - *Purpose*: Detailed XAS spectra for expert users, where one needs to do complex/advanced analysis.
  - *Output*: As in `ABSQ` process all avalable tables including:
  1. **Origin-dependent tables**
  2. **Origin-adjusted tables** [670].
  3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].
  - *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.

- **XASSOC** (SOC X-ray Absorption Spectra)
  - *Description*: `XAS` with spin-orbit coupling.
  - *Purpose*: Processing spectra generated on the basis of X-ray ABSORPTION upon core electron ioniztion including SOC effects.
  - *Use Case*: Recommended for processing routine SOC-XAS spectra.
  - *Output*: Absorption spectrum with SOC effects (Intensity (oscillator strenth) vs. energy (e.g. electron volt))

- **XASSOCV** (SOC X-ray Absorption Spectra in Velocity Representation)
  - *Description*: `XASSOC` in velocity representation.
  - *Purpose*: Alternative formulation for `XASSOC`.
  - *Output*: Similar to `XASSOC`.

- **XASSOCFFMIO** (SOC X-ray Absorption Spectra with Full Matter Interaction Operator)
  - *Description*: `XASSOC` employing beyond dipole approximation intensity contributions.
  - *Purpose*: Recommended for processing routine SOC-XAS spectra beyond dipole approximation.
  - *Output*: X-ray absorption beyond dipole approximation of FFMIO operator including SOC corrections (Intensity (oscillator strenth) vs. energy (e.g. electron volt))

- **XASSOCQ** (SOC X-ray Absorption Spectra with Dipole, Quadrupole, and Magnetic Dipole)
  - *Description*: `XASSOC` Including electric dipole (length), electric quadrupole, and magnetic dipole contributions multipolar formulations beyond dipole.
  - *Purpose*: Detailed XASSOC spectra for expert users, where one needs to do complex/advanced analysis.
  - *Output*: As in `SOCABSQ` process all avalable tables including:
  1. **Origin-dependent tables**
  2. **Origin-adjusted tables** [670].
  3. **Origin-independent tables** using corrected higher moments from second-order terms in the multipole expansion [647].
  - *Notes*: Produces comprehensive data for advanced/expert analysis provided that the **DecomposeFosc** has been requested in the input run.

Resonant Inelastic X-ray Scattering (RIXS)

These keywords process RIXS spectra generated by a variety of methods

- **RIXS** (Resonant Inelastic X-ray Scattering)
  - *Description*: Processes RIXS spectra (energy loss of scattered X-rays, along incident and Emission/Energy Loss Axis).
  - *Purpose*: Probes electronic and vibrational structure.
  - *Use Case*: Transition metals, catalysts.
  - *Output*: 2D spectrum as well as 1D profile spectra along constant Emission/Incident Axis.
  - *Notes*: See section *Resonant Inelastic Scattering Spectroscopy*
- **RIXSSOC** (SOC Resonant Inelastic X-ray Scattering)
  - *Description*: `RIXS` with spin-orbit coupling.
  - *Purpose*: Accounts for SOC in RIXS spectra.
  - *Use Case*: RIXS for chemical systems with significant multiple structure (2p3d RIXS).
  - *Output*: 2D spectrum as well as 1D profile spectra along constant Emission/Incident Axis with SOC effects.
  - *Notes*: See section *Resonant Inelastic Scattering Spectroscopy*

Transient Spectroscopy

These keywords compute time-resolved spectra for pump-probe experiments.

- **TRANSABS** (Transient Absorption Spectra)
  - *Description*: Processing transient absorption spectra.
  - *Purpose*: Models time-dependent absorption changes.
  - *Use Case*: Photochemistry or Time-resolved or Pump-probe UV absorption studies.
  - *Output*: Transient absorption spectrum, (Intensity vs. energy).
- **TRANSCD** (Transient Circular Dichroism)
  - *Description*: Processing transient CD spectra.
  - *Purpose*: Models time-dependent CD changes.
  - *Use Case*: Time-resolved or Pump-probe CD spectra in chiral systems.
  - *Output*: $(\Delta\epsilon)$ vs. energy.
- **TRANSSOCABS** (SOC Transient Absorption Spectra)
  - *Description*: `TRANSABS` with spin-orbit coupling.
  - *Purpose*: Accounts for SOC in transient absorption.
  - *Use Case*: Time-resolved or Pump-probe ABS/XAS/XUV for chemical systems with complex multiple structure
  - *Output*: Transient absorption spectrum with SOC (Intensity vs. energy)

Vibrational Spectroscopy

These keywords compute spectra related to vibrational transitions.

- **IR** (Infrared Spectroscopy)
  - *Description*: Processing infrared absorption spectra.
  - *Purpose*: Models vibrational transitions.
  - *Use Case*: Molecular vibrational analysis/Structural analysis.
  - *Output*: Intensity vs. wavenumber.
- **VCD** (Vibrational Circular Dichroism)
  - *Description*: Processing vibrational CD spectra.
  - *Purpose*: Probes chiral vibrational transitions.
  - *Use Case*: Chiral molecules (e.g., biomolecules).
  - *Output*: $(\Delta\epsilon)$ vs. wavenumber.
- **RAMAN** (Raman Spectroscopy)
  - *Description*: Processing Raman scattering spectra.
  - *Purpose*: Models vibrational transitions via Raman scattering.
  - *Use Case*: Molecular vibrational analysis/Structural analysis.
  - *Output*: Intensity vs. wavenumber.
- **NRVS** (Nuclear Resonant Vibrational Spectroscopy)
  - *Description*: Processing nuclear resonant vibrational spectra.
  - *Purpose*: Probes vibrational modes coupled to nuclear transitions (e.g., Fe-57).
  - *Use Case*: Usefull in Bioinorganic chemistry analysis.
  - *Output*: Intensity vs. energy.
- **VDOS** (Vibrational Density of States)
  - *Description*: Processing vibrational density of states.
  - *Purpose*: Provides distribution of vibrational modes.
  - *Use Case*: Usefull in Solids or large systems analysis.
  - *Output*: Density vs. energy.

General Notes

- *Dipole vs. Velocity*: Length (`ABS`) and velocity (`ABSV`) representations are equivalent for exact wavefunctions but may differ in approximate calculations.
- *Beyond Dipole*: `FFMIO` and `Q` keywords include quadrupole and magnetic dipole terms, critical for X-ray spectroscopy.
- *Spin-Orbit Coupling*: `SOC` keywords are essential for heavy elements or spin-forbidden transitions.
- *Expert vs. Conventional*: `Q` variants produce detailed output for experts; `FFMIO` variants should be used for routine analysis.
- *ORCA Requirements*: Some keywords require specific settings (e.g., relativistic methods for `SOC`, request `DOTRANS true/all` for transient Absorption methods).

Example Usage

NOTE:

- The input to the `orca_mapspc` program can be either a standard ORCA output file or .hess file for vibrational spectra such as IR and Raman.
- Unless specified otherwise, the line shape is assumed to be Gaussian.
- The `orca_mapspc` program generates two output files:
  - `Input-file.spc.dat` (spc = abs, cd, ir, or raman): This file contains the data required to plot the continuous band-shape spectrum.
  - `Input-file.spc.stk`: This file contains the individual transitions (wavenumber and intensity) to plot the line spectrum.
- In the case of absorption spectra, the .dat file contains five columns: wavenumber in reciprocal centimeters, total intensity, polarization components along the -x, y-, and z-axes of the electric field vector of the incoming beam. The polarization components are especially useful for interpreting polarized single crystal spectra.
- When more than one spectra of the same kind is present in the ORCA output file, `orca_mapspc` will automatically process all these spectral datasets. For example, if the ORCA output includes both CASSCF and NEVPT2 excitations, spectral data corresponding to both methods will be processed. To illustrate this, assume the CASSCF and NEVPT2 absorption data are both oresent in an ORCA output file named My-NEVPT2.out. Let us pass this file to `orca_mapspc`:

```
orca_mapspc My-NEVPT2.out SOCABS -x07000 -x18000 -eV -n10000 -w2.0 -l
```

`orca_mapspc` will then print out:

```
Mode is SOCABS
Entering SOC-ABS reading
Using eV units
Using Lorentzian shape
Multiple SOCABS (2) spectra detected ...
----------------------------------
Plotting SOCABS Spectrum 0
----------------------------------
Cannot read the paras.inp file ...
taking the line width parameter from the command line
Number of peaks            ...   4455
Start energy [eV]          ...  7000.00
Stop energy [eV]           ...  8000.00
Peak FWHM [eV]             ...     2.00
Number of points           ...   10000
----------------------------------
Plotting SOCABS Spectrum 1
----------------------------------
Cannot read the paras.inp file ...
taking the line width parameter from the command line
Number of peaks            ...   4455
Start energy [eV]          ...  7000.00
Stop energy [eV]           ...  8000.00
Peak FWHM [eV]             ...     2.00
Number of points           ...   10000
```

Finally, it will produce data files for both CASSCF and NEVPT2 flags:

```
CASSCF:
       My-NEVPT2.out.0.socabs.dat
       My-NEVPT2.out.0.socabs.stk
NEVPT2:
```

(continues on next page)

(continued from previous page)

```
        My-NEVPT2.out.1.socabs.dat
        My-NEVPT2.out.1.socabs.stk
```

Other available spectral data in the My-NEVPT2.out file, such as CD spectra, can also be processed in the same way.

9.2.2 orca_chelpg

The `orca_chelpg` program calculates CHELPG atomic charges according to Breneman and Wiberg[582]. The atomic charges are fitted to reproduce the electrostatic potential (ESP) on a regular grid surrounding the molecule, while constraining the sum of all atomic charges to the molecule's total charge. More information on the defaults is available in *CHELPG Charges*.

In order to run the program from the terminal, only one argument needs to be provided; the gbw filename:

```
orca_chelpg  MyJob.gbw
```

as an optional argument one can provide the total solute density matrix via a file name.

9.2.3 orca_pltvib

The `orca_pltvib` program is used in conjunction with gOpenMol (or xmol) to generate animations or plots of vibrational modes following a frequency calculation. Its usage is straightforward and is described in the section *Animation of Vibrational Modes*, along with a brief guide on how to produce these plots in gOpenMol.

The program generates 20 animation frames per vibrational mode. The first and last frames correspond to the TS, while the intermediate frames are computed as $sin(2\pi frame/20 - 1) * displacement$. This produces a smooth oscillation pattern centered on the equilibrium geometry.

9.2.4 orca_vib

`orca_vib` is a small "standalone" program for performing vibrational analyses. It gives users control over parameters such as atomic masses, which influence the predicted vibrational frequencies but are independent of the electronic structure calculation itself.

The program takes a "`.hess`" file as input and generates output that is essentially equivalent to that produced by a standard ORCA frequency calculation. The key advantage is that the `.hess` file is a user-editable plain text file, which allows manual modifications, e.g., for isotope shift predictions or custom mass adjustments.

Usage of `orca_vib`, along with an example, is provided in the section *Isotope Shifts*. By piping the screen output to a text file, you can further process the vibrational data using `orca_mapspc` to generate data files for plotting IR, Raman, or NRVS spectra from the modified Hessian.

9.2.5 orca_loc

Localization is a widely used technique in quantum chemistry. By defining different functionals, various localization methods have been developed. Among the most widely used are the Foster–Boys (FB) and Pipek–Mezey (PM) methods. In ORCA, six localization methods are available: Pipek-Mezey method (PM), Foster-Boys method (FB), intrinsic atomic orbitals (IAO)-based PM method, IAO-based FB method, PM loclization of valence virtual orbitals (PMVVO), and Localized intrinsic valence virtual orbitals (LIVVO).

Three algorithms are available for FB localizaton: the conventional algorithm (`FB`), a faster alternative `NEWBOYS`, recommended for localizing virtual MOs in large systems, and augmented Hessian FB (`AHFB`) algorithm, ideal for obtaining tightly converged orbitals (e.g., for local correlation methods with an appropriate tolerance).

The `AHFB` algorithm, which systematically converges towards a local minimum, rather than a different type of stationary point, proceeds in three stages: (i) An initial set of `NEWBOYS` localized orbitals is obtained. (ii) This is

followed by an augmented Hessian maximization (rational function optimization) using direct or Davidson diagonalization, depending on the number of orbitals to make it efficient for both small and large systems. (iii) If the optimization fails but the augmented Hessian has the correct eigenvalue structure, a Newton-Raphson maximization is triggered.

The only user-adjustable parameter in AHFB is the tolerance Tol. Convergence is signalled when the eigenvalue structure is correct, and the largest element of the orbital gradient, $4\langle i|\mathbf{r}|j\rangle\left(\langle j|\mathbf{r}|j\rangle - \langle i|\mathbf{r}|i\rangle\right)$, is below Tol. This differs from other localization methods, which use the sum of changes in the localization functional between two successive iterations as the convergence criterion.

The intrinsic atomic orbitals and intrinsic bond orbitals (IAOIBO) localization method, developed by Gerald Knizia [820], proceeds as follows: (i) The occupied MOs are projected onto a minimal basis set to get the IAOs. In ORCA, IAOs are generated from the converged SCF MO of atoms by default, instead of the MINI or STO-3G basis sets as in the original method. However, the IAO charges computed by ORCA default are very similar to the original IAO charges. The default behaviour can be changed using the IAOBasis keyword. (ii) The PM functional is used to localize these IAOs into IBOs. (iii) The IBOs are back-transformed to their original basis set.

NOTE: IAO partial charges for canonical MOs are printed before the IAOIBO localization. Be sure to include all occupied MOs, as missing any will render the IAO charges meaningless.

An improved version of this approach in ORCA replaces PM with FB as the localization functional, resulting in the IAOBOYS method. For large systems, IAOBOYS is computationally more efficient than the standard FB method. However, it is worth noting that IAO-based localization is currently only available for occupied MOs.

Valence virtual orbitals (VVOs) can be localized following a method described in ref. [821]. Thus method can be briefly described as follows: The VVOs are obtained via singular value decomposition of the overlap between IAOs and virtual MOs. The difference between the number of IAOs and occupied MOs is the number of VVOs. However, the full virtual orbital window must be provided as input to orca_loc! The final window of VVOs is printed in the output. The remaining virtual MOs are not modified and thus not orthogonal to the VVOs. The VVOs can then be localized using the PM or IBO criterion using the PMVVO and LIVVO keyword, respectively. The latter option corresponds to the localized intrinsic VVOs of ref. [821].

There are two ways to request MO localization in ORCA: (i) Inline in the input file using the %loc block and (ii) using the orca_loc utility on precomputed MOs.

When requested in the %loc block of the input file, the following options can be set:

```
%loc
LocMet       PM            # Localization method e.g. PIPEK-MEZEY
             FB            # FOSTER-BOYS
             IAOIBO        # IAOIBO
             IAOBOYS       # IAOBOYS
             NEWBOYS       # FOSTER-BOYS
             AHFB          # Augmented Hessian Foster-Boys
             PMVVO         # PM-localized valence virtual orbitals
             LIVVO         # localized intrinsic valence virtual orbitals
LocMetVirt   0             # If given, used for the virtual space instead of LocMet;
                           # the same options are available
Tol          1e-6          # absolute convergence tolerance for the localization sum
                           # default value is 1e-6
                           # In the case of AHFB, however, this is the gradient␣
↪threshold!
Random       0             # Always take the same seed for start for localization
                           # (For testing/debug purpose,optional)
             1             # Take a random seed for start of localization (default)
PopMethod    MULLIKEN      # Population method for the final analysis
             LOEWDIN       # of the localized orbitals (default: Mulliken)
PrintLevel   2             # Amount of printing
OrbSpread    false         # Whether to compute and print the localized orbital␣
↪spread:
                           # sqrt(<i|(r-<i|r|i>)^2|i>)
MaxIter      64            # Max number of iterations
T_Bond       0.85          # Thresh that classifies orbitals in bond-like at the␣
```

(continues on next page)

(continued from previous page)

```
↪printing
T_Strong    0.95          # Thresh that classifies orbitals into strongly-
↪localized at
                          # the printing
OCC         true          # Localize the occupied space
T_CORE      -99.9         # The Energy window for the first OCC MO to be localized␣
↪(in a.u.)
                          # Here, we localize all occupied MOs including core␣
↪orbitals.
VIRT        true          # Localize the virtual space
IAOBasis    SCF_SV        # (default) Basis of MOs obtained from HF/SV calculations
            STO_3G        # STO-3G
            MINI          # MINI
            ANO_SZ        # ANO-SZ
            ANO_RCC_MB    # ANO-RCC-MB
            MINAO_AUTO_PP # MINAO-AUTO-PP basis (presumes ECPs beyond Kr)
end
```

MOs are localized in ORCA iteratively. Convergence is achieved when the localization functional value becomes self-consistent, as controlled by the Tol parameter. If the flag `OCC` or `VIRT` is set to true in the %loc block, the localization is performed for the corresponding occupied or virtual orbital subspace. If both `OCC` and `VIRT` flags are set to true, two consecutive localizations are carried out—one for each subspace. The localization method used for virtual orbitals is specified via the `LocMetVirt` keyword (if provided); otherwise, it defaults to `LocMet`. The localized orbitals are stored in a standard `.gbw` file format, but with the extension `.loc.gbw`.

NOTE: The use of localized occupied orbitals may alter the total energy. For RHF and UHF, this typically does not cause issues. However, for CASSCF, the OCC flag alone is not sufficient, because CASSCF is not invariant under rotations of all occupied orbitals.

When `orca_loc` program is called from the command line, it requires a separate text-based input file containing all necessary parameters. If no input is specified, `orca_loc` returns a help message listing all available options. Typically one needs to specify in the `orca_loc` input the input and output gbw fies, along with orbital ranges and the localization method to be used. A source of confusion is the operator line op (alpha = 0 or beta = 1). For RHF(ROHF) and CASSCF, this should be set to zero. The input file of `orca_loc` looks as follows:

```
Myjob.gbw     # input orbitals
Myjob.loc.gbw # output orbitals
10            # orbital window: first orbital to be localized e.g. first active
15            # orbital window: last orbital to be localized e.g. last active
0             # localization method:
              # 1=PIPEK-MEZEY,2=FOSTER-BOYS,3=IAO-IBO,4=IAO-BOYS,5=NEW-BOYS,6=AHFB,
↪7=PMVVO,8=LIVVO
# The following parameters are optional
# However, if you want to change one of them, all preceding ones have to be set,␣
↪too.
0             # operator: 0 for alpha, 1 for beta
128           # maximum number of iterations
1e-6          # convergence tolerance of the localization functional value
0.0           # relative convergence tolerance of the localization functional value
0.95          # printing thresh to call an orbital strongly localized
0.85          # printing thresh to call an orbital bond-like
2             # printlevel
1             # use Cholesky Decomposition (0=false, 1=true)
1             # randomize seed for localization (0=false, 1=true)
0             # IAO basis (0-5 in the order given above)
0             # Population analysis method (0=Mulliken, 1=Lowedin)
0             # Print orbital spread (0=false, 1=true)
```

If this input file is named as myloc.inp, running "orca_loc myloc.inp" will produce the Myjob.loc.gbw file containing the localized orbitals. Please make sure the Myjob.gbw is in the same directory as myloc.inp.

9.2.6 orca_blockf

The `orca_blockf` utility program canonicalizes orbitals within a `.gbw` file for arbitrary subspaces. In this context, canonicalization refers to the block-diagonalization of the Fock matrix within the specified orbital subspace.

To run `orca_blockf` on the command line:

```
orca_blockf Fock.gbw Orbitals.gbw Output.gbw firstOrb lastOrb
```

Usage requires two `.gbw` files. One from which the Fock matrix will be reconstructed (Fock.gbw in the above) and another with the orbitals to be recanonicalized (Orbitals.gbw in the above). If the user doesn't provide an output `gbw` name, the program will overwrite the second argument `gbw` file.

This program is further described in section *Local Zero-Field Splitting*.

9.2.7 orca_plot

`orca_plot` produces plot-ready files (such as `.plt` and `.cube`) for a range of orbital and density types computed with ORCA, enabling visualization with external 2D and 3D graphics software. It can be called in two ways:

1) within an ORCA input file via the `%plots` block. This approach allows you to generate graphical data (e.g., .plt or .cube files) automatically during the run. Details are provided in section *Orbital and Density Plots*.

2) interactively after a calculation, using the already generated `.gbw` file with following available parameters:

```
gbwfile     # name of gbw-file
-i          # interactive use of orca_plot
-m 256      # max. memory in MB (if needed)
```

If needed, the -m-option can be used to control the memory usage of the plotting job. `orca_plot` can be called in the terminal, for example, for the gbw file named my.gbw, as:

```
orca_plot my.gbw -i
```

This will list the available options:

```
1 - Enter type of plot
2 - Enter no of orbital to plot
3 - Enter operator of orbital (0=alpha,1=beta)
4 - Enter number of grid intervals
5 - Select output file format
6 - Plot CIS/TD-DFT difference densities
7 - Plot CIS/TD-DFT transition densities
8 - Set AO(=1) vs MO(=0) to plot
9 - List all available densities
10 - Perform Density Algebraic Operations

11 - Generate the plot
12 - exit this program
```

In the following, let us demonstrate the use of `orca_plot` for different plot types using the above option list.

Perform Orbital Plots

Let us consider an ORCA input file for a single-point energy calculation on the pyridine molecule:

```
! RHF def2-SVP

*xyz 0 1
C        0.690940233      0.417992301     -1.170801378
C        0.690940233      1.616339301     -0.458357378
C        0.690940233      1.560238301      0.936438622
N        0.690940233      0.417992301      1.635468622
C        0.690940233     -0.724253699      0.936438622
C        0.690940233     -0.780354699     -0.458357378
H        0.690940233      0.417992301     -2.257043378
H        0.690940233      2.574997301     -0.967574378
H        0.690940233      2.478336301      1.521602622
H        0.690940233     -1.642351699      1.521602622
H        0.690940233     -1.739012699     -0.967574378
*
```

We want to plot the HOMO orbital. First, we need to determine its orbital number (20) from the `ORBITAL ENERGIES` section of the ORCA output file:

```
----------------
ORBITAL ENERGIES
----------------

 NO   OCC          E(Eh)            E(eV)
  0   2.0000     -15.563726      -423.5105
...
Occupied Orbitals Manifold
...
 20   2.0000      -0.349834        -9.5195
...
Unoccupied Orbitals Manifold
...
 21   0.0000       0.111722         3.0401
...
```

Then let us specify appropriate choices for options `2`, `3`, `4`, `5` and `8`:

```
Enter a number: 2
Enter MO: 20

Enter a number: 3
Enter OP: 0

Enter a number: 4
Enter NGRID: 80

Enter a number: 5
File-Format is presently: 7
(7 -  3D   Gaussian cube)

Enter a number: 8
Enter 0(MO) or 1(AO): 0
```

To generate the plot, choose `11`. Then, exit `orca_plot` with choose `12` if no additional plots are needed. Otherwise, multiple plots can be requested sequentially in a single run.

```
11 - Generate the plot
Enter a number: 11 =>
```

(continues on next page)

(continued from previous page)

```
PlotType        ... MO-PLOT
MO/Operator     ... 20 0
Output file     ... pyridine_scf.mo20a.cube
Format          ... Grid3d/Cube
Resolution      ... 80 80 80

Calling PlotGrid3d with ATOM-A,B=0,0
Entering PlotGrid3d with Plottype =1
                 *** PLOTTING FINISHED ***
Output file: pyridine_scf.mo20a.cube
```

This generates a `.cube` file that can be used to visualize the HOMO orbital with a visualization program. The picture of the HOMO orbital obtained using this cube file as input in `Chimera` is shown in Fig. 9.4.

Fig. 9.4: Pyridine HOMO

List of Density Plots

If a density instead of an orbital plot is required options `1` and `9` can be used to list the available densities.

For example option `1` will provide the following computed available densities for the above pyridine example.

```
Enter a number: 1
-----------------------------------------------------------------------
Reading Over   2 Saved Densities                                    ...
-----------------------------------------------------------------------
-----------------------------------------------------------------------
Plot-Type is presently: 1
-----------------------------------------------------------------------
Searching for Ground State Electron or Spin Densities:              ...
-----------------------------------------------------------------------
     1 -   molecular orbitals
     2 -   (scf) electron density                              ......  (scfp      ␣
↪              )  => AVAILABLE
     3 -   (scf) spin density                                  ......  (scfr      ␣
↪              )  - NOT AVAILABLE
     4 -   natural orbitals
```

(continues on next page)

(continued from previous page)

```
     5 -   corresponding orbitals
     6 -   atomic orbitals
     7 -   mdci electron density                              ......  (mdcip     ␣
↪              )  - NOT AVAILABLE
     8 -   mdci spin density                                  ......  (mdcir     ␣
↪              )  - NOT AVAILABLE
     9 -   OO-RI-MP2 density                                  ......  (pmp2re    ␣
↪              )  - NOT AVAILABLE
    10 -   OO-RI-MP2 spin density                             ......  (pmp2ur    ␣
↪              )  - NOT AVAILABLE
    11 -   MP2 relaxed density                                ......  (pmp2re    ␣
↪              )  - NOT AVAILABLE
    12 -   MP2 unrelaxed density                              ......  (pmp2ur    ␣
↪              )  - NOT AVAILABLE
    13 -   MP2 relaxed spin density                           ......  (rmp2re    ␣
↪              )  - NOT AVAILABLE
    14 -   MP2 unrelaxed spin density                         ......  (rmp2ur    ␣
↪              )  - NOT AVAILABLE
    15 -   LED dispersion interaction density                 ......  (ded21     ␣
↪              )  - NOT AVAILABLE
    16 -   Atom pair density
    17 -   Shielding Tensors
    18 -   Polarisability Tensor
    19 -   AutoCI relaxed density                             ......  (autocipre ␣
↪              )  - NOT AVAILABLE
    20 -   AutoCI unrelaxed density                           ......  (autocipur ␣
↪              )  - NOT AVAILABLE
    21 -   AutoCI relaxed spin density                        ......  (autocirre ␣
↪              )  - NOT AVAILABLE
    22 -   AutoCI unrelaxed spin density                      ......  (autocirur ␣
↪              )  - NOT AVAILABLE
```

In this case, only the SCF electron density (option `2`) is available, and can thus be prosessed using `orca_plot` to prepare an input file for visualization software. Choose this option to generate the necessary `scfp` file:

```
Enter Type: 2
The default name of the density would be: pyridine_scf.scfp
Is this the one you want (y/n)?
```

The generated file will then appear under option `9` in the list of available density files.

```
---------------------
List of density names
---------------------

Index:                                                    Name of Density
-----------------------------------------------------------------------
    0:                                                  pyridine_scf.scfp
```

Following the steps described for generating the cube file of the HOMO orbital, a file (cube, plt, etc.) necessary for visualizing the SCF electron density of pyridine can also be generated. When option `11` is selected, the following will be printed out in the terminal.

```
PlotType        ... DENSITY-PLOT
ElDens File     ... pyridine_scf.scfp
Output file     ... MyElDens
Format          ... Grid3d/Cube
Resolution      ... 80 80 80

Calling PlotGrid3d with ATOM-A,B=0,0
Entering PlotGrid3d with Plottype =2
```

(continues on next page)

(continued from previous page)

```
                 *** PLOTTING FINISHED ***
 Output file: pyridine_scf.eldens.cube
```

The plot of this SCF electron density using the generated cube file as input to Chimera is as in Fig. 9.5.

Fig. 9.5: Pyridine SCF Electron Density

Perform Algebraic Operations

Starting from ORCA 6, one can perform simple algebraic operations with the computed densities. The presently available operations are listed in menu option `10`:

```
Enter a number: 10
-----------------------------------------------------------------------
Available Algebraic Operations:
-----------------------------------------------------------------------
       1 - Pair Densities Addition
       2 - Pair Densities Subtraction
```

(continues on next page)

(continued from previous page)

```
       3 - Pair Densities Multiplication
       4 - Pair Densities Division
       5 - Density Normalization
       6 - Make Natural Transition Orbitals
       7 - Make Natural Difference Orbitals
       8 - Leave Section - Return to Main Menu
```

To perform such operations, the desired state or transition-state densities must be available in the `.densities` file. This can be achieved by requesting the relevant densities to be written to disk using the `!KeepDens` or `!KeepTransDensity` keywords in the input.

NOTE: Storing hundreds or thousands of such densities requires care, as they may occupy hundreds of gigabytes of disk space.

In addition to the SCF calculation on pyridine, let us now perform a canonical CCSD calculation with the following input:

```
! CCSD def2-SVP KeepDens

*xyz 0 1
C        0.690940233      0.417992301     -1.170801378
C        0.690940233      1.616339301     -0.458357378
C        0.690940233      1.560238301      0.936438622
N        0.690940233      0.417992301      1.635468622
C        0.690940233     -0.724253699      0.936438622
C        0.690940233     -0.780354699     -0.458357378
H        0.690940233      0.417992301     -2.257043378
H        0.690940233      2.574997301     -0.967574378
H        0.690940233      2.478336301      1.521602622
H        0.690940233     -1.642351699      1.521602622
H        0.690940233     -1.739012699     -0.967574378
*
```

Option `1` in the `orca_plot` menu will now provide both the scf and mdci electron densities.

```
----------------------------------------------------------------------
Plot-Type is presently: 1
----------------------------------------------------------------------
Searching for Ground State Electron or Spin Densities:              ...
----------------------------------------------------------------------
     1 -   molecular orbitals
     2 -   (scf) electron density                              ......  (scfp      ␣
↪              )  => AVAILABLE
     3 -   (scf) spin density                                  ......  (scfr      ␣
↪              )  - NOT AVAILABLE
     4 -   natural orbitals
     5 -   corresponding orbitals
     6 -   atomic orbitals
     7 -   mdci electron density                               ......  (mdcip     ␣
↪              )  => AVAILABLE
```

Let us compute correlation electron density by taking the difference of CCSD and HF electron densities. Then, we need to choose option `2` (Pair Densities Subtraction) of the `Available Algebraic Operations` menu:

```
---------------------
List of density names
---------------------

Index:                                                  Name of Density
----------------------------------------------------------------------
    0:                                                pyridine_ccsd.scfp
```

(continues on next page)

(continued from previous page)

```
    1:                                                  pyridine_ccsd.P0.tmp
    2:                                                   pyridine_ccsd.mdcip


----------------------------------------------------------------------
Performing Algebraic Operations Over Densities: => SUBTRACTION
----------------------------------------------------------------------
Number of Densities to be Processed from the List =>   2:
----------------------------------------------------------------------
Enter FileName for Density[  0]: pyridine_ccsd.scfp
Provide a Scale Factor for Density[  0] (Default => 1.00): 1
Enter FileName for Density[  1]: pyridine_ccsd.mdcip
Provide a Scale Factor for Density[  1] (Default => 1.00): 1
----------------------------------------------------------------------
INTERPRETTING EQUATION:
----------------------------------------------------------------------
1/sqrt(N) * [(1.00) * {pyridine_ccsd.scfp} - (1.00) * {pyridine_ccsd.mdcip}]
----------------------------------------------------------------------

PlotType       ... DENSITY-PLOT
ElDens File    ... pyridine_ccsd.scfp_minus_pyridine_ccsd.mdcip
Output file    ... pyridine_ccsd.scfp_minus_pyridine_ccsd.mdcip.cube
Format         ... Grid3d/Cube
Resolution     ... 80 80 80
```

The generated `pyridine_ccsd.scfp_minus_pyridine_ccsd.mdcip.cube` file can be directly visualized with a software like Chimera (see Fig. 9.6).

Fig. 9.6: Pyridine CCSD-HF Electron Density

NOTE: The generated density is stored in the Density Container so that it can be further processed or properly stored for future use. Now when option `9` is selected, the difference density will be seen among the available densities.

```
---------------------
List of density names
---------------------

Index:                                                  Name of Density
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------
    0:                                                pyridine_ccsd.scfp
    1:                                              pyridine_ccsd.P0.tmp
    2:                                               pyridine_ccsd.mdcip
    3:                        pyridine_ccsd.scfp_minus_pyridine_ccsd.mdcip
```

Plot Natural Transition Orbitals (NTOs) and Natural Difference Orbitals (NDOs)

`orca_plot` can also be used to generate Natural Transition Orbitals (NTOs) and Natural Difference Orbitals (NDOs) from any available state or transition density, regardless of the level of theory used. Let us now demonstrate this on the pyridine molecule for a state-averaged CASSCF(7,8) calculation. For this analysis, the relevant densities need to be stored to disk by including the `!KeepTransDensity` keyword in the input line. A corresponding ORCA input is as follows:

```
! def2-SVP KeepTransDensity

%casscf
 nel 8
 norb 7
 mult 1
 nroots 10
end


*xyz 0 1
C        0.690940233      0.417992301     -1.170801378
C        0.690940233      1.616339301     -0.458357378
C        0.690940233      1.560238301      0.936438622
N        0.690940233      0.417992301      1.635468622
C        0.690940233     -0.724253699      0.936438622
C        0.690940233     -0.780354699     -0.458357378
H        0.690940233      0.417992301     -2.257043378
H        0.690940233      2.574997301     -0.967574378
H        0.690940233      2.478336301      1.521602622
H        0.690940233     -1.642351699      1.521602622
H        0.690940233     -1.739012699     -0.967574378
*
```

When this calculation is run, in addition to the CASSCF electron density, all relevant CASSCF state and transition densities become available, as can be seen by choosing option `1` in the main `orca_plot` menu:

```
-----------------------------------------------------------------------
Plot-Type is presently: 1
-----------------------------------------------------------------------
Searching for Ground State Electron or Spin Densities:              ...
-----------------------------------------------------------------------
     1 -   molecular orbitals
     2 -   (scf) electron density                              ......  (scfp      ␣
↪              )  => AVAILABLE
     3 -   (scf) spin density                                  ......  (scfr      ␣
↪              )  - NOT AVAILABLE
     ...
-----------------------------------------------------------------------
Searching for State or Transition State AO Electron Densities:      ...
-----------------------------------------------------------------------
    23 -   CIS unrelaxed transition AO density                 ......  (Tdens-CIS ␣
↪              )  - NOT AVAILABLE
    24 -   ROCIS unrelaxed transition AO density               ......  (Tdens-
↪ROCIS              )  - NOT AVAILABLE
```

(continues on next page)

(continued from previous page)

```
    25 -   CAS unrelaxed transition AO density                 ......  (Tdens-CAS ␣
↪              )  => AVAILABLE
    ...
-----------------------------------------------------------------------
Searching for State or Transition State MO Electron Densities:      ...
-----------------------------------------------------------------------
    29 -   CIS unrelaxed transition MO density                 ......  (Tdens-
↪CISMO              )  - NOT AVAILABLE
    30 -   ROCIS unrelaxed transition MO density               ......  (Tdens-
↪ROCISMO            )  - NOT AVAILABLE
    31 -   CAS unrelaxed transition MO density                 ......  (Tdens-
↪CASMO              )  => AVAILABLE
   ...
```

We can now proceed to generate the NTOs and NDOs that dominate the computed State 1.

```
ROOT   1:  E=    -246.3609192216 Eh  5.176 eV  41749.8 cm**-1
      0.82391 [   346]: 2212100
      0.06447 [   295]: 2112200
```

With options `6 =>NTOs` of the `Available Algebraic Operations` menu, one finds AO transition density D_{01} file for NTOs as:

```
-----------------------------------------------------------------------
Performing Algebraic Operations Over Densities: => MAKE_NTOS
-----------------------------------------------------------------------
Enter FileName for Density[  0]: Tdens-CAS-0-0-0-1
Provide a Scale Factor for Density[  0] (Default => 1.00): 1
-----------------------------------------------------------------------
NATURAL TRANSITION ORBITALS GENERATION:
-----------------------------------------------------------------------
Warning: The one-electron matrix doesn't exist - is recalculated (SHARK)
Calculating the overlap matrix            ... done!

-------------------------------------------------
NATURAL TRANSITION ORBITALS FOR STATE     1 1A
-------------------------------------------------

STATE    1 1A  :  E=   0.190226 au      5.176 eV    41749.8 cm**-1

Threshold for printing occupation numbers 1.0000e-04

     0  : n=  1.28519446
     1  : n=  0.03997952
     2  : n=  0.01505089
     3  : n=  0.00336119

=> Natural Transition Orbitals (donor   ) were saved in Tdens-CAS-0-0-0-1.1-1A_nto-
↪donor.gbw
=> Natural Transition Orbitals (acceptor) were saved in Tdens-CAS-0-0-0-1.1-1A_nto-
↪acceptor.gbw
-----------------------------------------------------------------------
Provide a Number of NTOs to plot (Default => 1): 1
-----------------------------------------------------------------------
-----------------------------------------------------------------------
Reading Donor NTO-file   : Tdens-CAS-0-0-0-1.1-1A_nto-donor.gbw
-----------------------------------------------------------------------
Generating cube file for Donor NTO[0]:=> Tdens-CAS-0-0-0-1.1-1A_nto-donor.gbw.0.
↪cube
-----------------------------------------------------------------------
Reading Acceptor NTO-file   : Tdens-CAS-0-0-0-1.1-1A_nto-acceptor.gbw
```

(continues on next page)

(continued from previous page)

```
-----------------------------------------------------------------------
Generating cube file for Acceptor NTO[0]:=> Tdens-CAS-0-0-0-1.1-1A_nto-acceptor.
↪gbw.0.cube
Current-settings:

PlotType        ... MO-PLOT
MO/Operator     ... 0 0
Output file     ... Tdens-CAS-0-0-0-1.1-1A_nto-acceptor.gbw.0.cube
Format          ... Grid3d/Cube
Resolution      ... 80 80 80
```

With options `7 =>NDOs` of the `Available Algebraic Operations` menu, one finds AO state density $D_{00} - D_{11}$ file for NDOs as:

```
-----------------------------------------------------------------------
Performing Algebraic Operations Over Densities: => MAKE_NDOS
-----------------------------------------------------------------------
Enter FileName for Density[  0]: Tdens-CAS-0-0-0-0
Provide a Scale Factor for Density[  0] (Default => 1.00): 1
Enter FileName for Density[  1]: Tdens-CAS-0-0-1-1
Provide a Scale Factor for Density[  1] (Default => 1.00): 1
-----------------------------------------------------------------------
NATURAL DIFFERENCE ORBITALS GENERATION:
-----------------------------------------------------------------------
Warning: The one-electron matrix doesn't exist - is recalculated (SHARK)
Calculating the overlap matrix            ... done!

-------------------------------------------------
NATURAL DIFFERENCE ORBITALS FOR STATE     1 1A
-------------------------------------------------

STATE    1 1A  :  E=   0.190226 au      5.176 eV    41749.8 cm**-1

Threshold for printing occupation numbers 1.0000e-04

     0  : n=  0.49881235
     1  : n=  0.08492368
     2  : n=  0.06708141
     3  : n=  0.01254920
     4  : n=  0.00667407
     5  : n=  0.00006162

=> Natural Difference Orbitals (donor   ) were saved in Tdens-CAS-0-0-0-0-Tdens-
↪CAS-0-0-1-1.1-1A_ndo-donor.gbw
=> Natural Difference Orbitals (acceptor) were saved in Tdens-CAS-0-0-0-0-Tdens-
↪CAS-0-0-1-1.1-1A_ndo-acceptor.gbw
-----------------------------------------------------------------------
Provide a Number of NDOs to plot (Default => 1): 1
-----------------------------------------------------------------------
-----------------------------------------------------------------------
Reading Donor NDO-file   : Tdens-CAS-0-0-0-0-Tdens-CAS-0-0-1-1.1-1A_ndo-donor.gbw
-----------------------------------------------------------------------
Generating cube file for Donor NDO[0]:=> Tdens-CAS-0-0-0-0-Tdens-CAS-0-0-1-1.1-1A_
↪ndo-donor.gbw.0.cube
-----------------------------------------------------------------------
Reading Acceptor NDO-file   : Tdens-CAS-0-0-0-0-Tdens-CAS-0-0-1-1.1-1A_ndo-
↪acceptor.gbw
-----------------------------------------------------------------------
Generating cube file for Acceptor NDO[0]:=> Tdens-CAS-0-0-0-0-Tdens-CAS-0-0-1-1.1-
↪1A_ndo-acceptor.gbw.0.cube
Current-settings:
```

(continues on next page)

(continued from previous page)

```
PlotType        ... MO-PLOT
MO/Operator     ... 0 0
Output file     ... Tdens-CAS-0-0-0-0-Tdens-CAS-0-0-1-1.1-1A_ndo-acceptor.gbw.0.cube
Format          ... Grid3d/Cube
Resolution      ... 80 80 80
```

Using the resulting cube file, one can readily visualize the corresponding Donor and Acceptor NTOs and NDOs orbital pairs as in which can be readily visualized in Fig. 9.7.

Fig. 9.7: Pyridine SA-CASSCF(8,7) NTO/NDO donor/acceptor orbital pairs for State 1

NOTE: It is both beneficial and more accurate to always process AO-basis densities. In particular, it is neither possible nor correct to process reduced MO-basis densities.

Perform Electrostatic Potential Plots

Starting with ORCA 6.1, the electrostatic potential (ESP) can be generated in `orca_plot` for any state density available in the density container.

After performing an electronic structure calculation (e.g., using *input*), proceed as follows in `orca_plot`:

First, choose `1 - Enter type of plot`, then select `43 - Electrostatic potential`.

```
    41 -   ROCIS QDPT unrelaxed transition AO density          ......  (Tdens-
↪ROCISQDSOC          )  - NOT AVAILABLE
    42 -   LFT QDPT unrelaxed transition AO density            ......  (Tdens-
↪LFTQDSOC            )  - NOT AVAILABLE
-----------------------------------------------------------------------
    43 -   Electrostatic Potential
```

You will be prompted to specify the name of the state density for which the electrostatic potential is to be computed.

```
Enter Type: 43

---------------------
List of density names
```

(continues on next page)

(continued from previous page)

```
---------------------

Index:                                                    Name of Density
-------------------------------------------------------------------------
    0:                                                  pyridine_scf.scfp


Enter Name for an STATE Density: pyridine_scf.scfp
```

Finally, select option `11 - Generate the plot` to compute the electrostatic potential and generate the file `pyridine_scf.scfp.esp.cube`.

The resulting `.cube` file contains the electrostatic potential data and may be visualized directly or used to color a density plot within any compatible graphical user interface tool.

Fig. 9.8: Pyridine SCF Electron Density using ESP colouring

Additional TIPS regarding Density Plots

1) As an alternative to menu option `9`, one can directly check which densities are available in the calculation directory by reading out the list of all densities contained in the `.densities` file:

```
orca_plot mymol.densities
```

2) `orca_plot` can also be run in parallel mode for computationally expensive plots:

```
mpirun -np 4 /orca_path/orca_plot_mpi MyGBWFile.gbw -i MyGBWFile
```

3) It is possible to use `orca_plot` to create difference densities between the ground and excited states from CIS or TD-DFT calculations directly from the .cis calculation information. This is implemented as an extra interactive menu point that is (hopefully) self-explanatory. Starting from ORCA 6 one can use the `Algebraic Operations` utility menu to carry out these operations directly from the generated State and Transition Densities.

4) It is also possible to use `orca_plot` to create difference densities between the ground and excited states from CIS or TD-DFT calculations using the `.cis` file. This functionality is implemented as an additional interactive menu option, which is (hopefully) self-explanatory. Starting from ORCA 6, these densities can also be obtained through the `Algebraic Operations` menu, processing directly the generated state and transition densities.

9.2.8 orca_2mkl: Old Molekel as well as Molden inputs

The `orca_2mkl` utility converts `.gbw` files into `.mkl` ASCII files. The `.mkl` file contains essentially the same information as the `.gbw` file, excluding ORCA-specific internal flags. The ASCII format makes it suitable for external use—for example, it can be read by Molekel for visualization or accessed by custom post-processing scripts. As such, `.mkl` files provide a convenient starting point for developing interfaces with third-party tools. `orca_2mkl` can be simply invoked as:

```
orca_2mkl BaseName
      (will produce BaseName.mkl from BaseName.gbw)
orca_2mkl BaseName -molden
      (writes a file in molden format)
orca_2mkl BaseName -mkl
      (writes a file in MKL format)
```

Any gbw-like file, containing such as QRO, UNO, UCO, etc. orbitals, can be converted to the mkl and Molden formats using `orca_2mkl`:

```
orca_2mkl gbw_type_file.extension mkl_file.extension -mkl -anyorbs
# or
orca_2mkl gbw_type_file.extension molden_file.extension -molden -anyorbs
```

`orca_2mkl` can also be run in reverse to generate `.gbw` type files from `.mkl` files:

```
orca_2mkl BaseName -gbw
      (will produce BaseName.gbw out of BaseName.mkl)
```

This functionality allows users to import molecular orbitals from external sources into ORCA. Although this feature is rarely used and offers limited capabilities, it is included here for users who wish to experiment. An example of this usage is demonstrated in the CASSCF tutorial accompanying the ORCA manual.

Finally, some third-party tools are available for converting the wavefunction files of other quantum chemistry programs from and to the MKL format, e.g. MOKIT (https://gitlab.com/jxzou/mokit). Combined with `orca_2mkl`, this provides a convenient way to use ORCA-generated molecular orbitals in other programs, or the other way around. One example application would be to use ORCA's TRAH method to converge a difficult SCF problem, and pass the converged orbitals to another program that does not have the TRAH converger. Another example is to read in the

orbitals of a difficult system from another program, and compute its high-level single point energy using ORCA using a functional or method that is not supported by the other program, without having to converge the wavefunction from the PModel guess. Finally, MOKIT can convert the MKL format to the PySCF format, which allows one to manipulate ORCA molecular orbitals in a Python environment.

9.2.9 orca_2aim

This utility program reads a `.gbw` file and creates a `.wfn` and `.wfx` file that can be used for topological analysis of the electron density by other programs. This works for open-shell and closed-shell wave functions. It can be invoked by including `! AIM` in the simple input line of the ORCA input file, or typing in the terminal as follows:

```
orca_2aim BaseName
      (will produce BaseName.wfn and BaseName.wfx from BaseName.gbw)
```

9.2.10 orca_vpot

This `orca_vpot` program calculates the electrostatic potential at a given set of user defined points. It can be used with either an input file or in interactive mode. It needs four arguments:

```
orca_vpot  MyJob.gbw  MyJob.scfp  MyJob.vpot.xyz  MyJob.vpot.out
```

First: The `gbw` file containing the correct geometry and basis set

Second: The desired density matrix in this basis (perhaps use the `KeepDens` keyword)

Third: an ASCII file with the target positions in AU, e.g.

```
6                  (number_of_points)
  5.0 0.0  0.0     (XYZ coordinates)
 -5.0 0.0  0.0
  0.0 5.0  0.0
  0.0-5.0  0.0
  0.0 0.0  5.0
  0.0 0.0 -5.0
```

Fourth: The target file which will then contain the electrostatic potential, e.g.

```
6                  (number of points)
 VX1  VY1  VZ1     (potential for first point)
 VX2  VY2  VZ2     (potential for second point)
  etc.
```

It is straightforward to read this file and use the potential for intended purpose.

`orca_vpot` can be called in parallel mode as follows:

```
mpirun -np 4 /full_path/orca_vpot_mpi MyJob.gbw MyJob.scfp MyJob.pot.xyz MyJob.pot.
↪out
```

It can be used to check available densities in MyJob.densities as:

```
orca_vpot  MyJob.densities
```

If the base names of the `.gbw` and `.densities` files do not match, the base name of the `.densities` file must be passed as the fifth argument:

```
orca_vpot  MyJob.gbw  OtherJob.scfp  MyJob.pot.xyz  MyJob.pot.out  OtherJob
```

A call to `orca_vpot` without any arguments will display a help message.

9.2.11 orca_euler

The `orca_euler` utility program calculates the relative orientation between calculated hyperfine coupling (HFC) or nuclear quadrupole coupling (NQC) tensors and a reference tensor,, typically the g-tensor or the D-tensor. The `orca_euler` program is invoked automatically at the end of an ORCA job if both the HFC/NQC and g-/D-tensor are calculated within the same run. Alternatively, it can be executed as a standalone program, provided that the .property.txt file from a previous NQC/HFC- and D- or g-tensor calculation is available.

The orientation between tensors is expressed as a 3×3 rotation matrix R, parameterized by the three Euler angles:α, β and γ. These angles define the relative orientation of tensor A with respect to tensor B through three successive rotations around different axes to align A with B, i.e., in the commonly used z-y-z convention:

- **Rotate $\mathbf{A}_{xyz}$ counterclockwise around its z axis by α to obtain $\mathbf{A}_{x'y'z'}$.**
- **Rotate $\mathbf{A}_{x'y'z'}$ counterclockwise around its y' axis by β to obtain $\mathbf{A}_{x''y''z''}$.**
- **Rotate $\mathbf{A}_{x''y''z''}$ counterclockwise around its z'' axis by γ to align it with B.**

The options of the `orca_euler` program are as follows:

```
orca_euler prop-file options

  file    = basename of an ORCA .property.txt file

  options
   -refg/-refD:         Reference tensor (g-tensor or D-tensor, default is -refg)
   -conv zyz/-conv zxz: Euler rotation convention (default is zyz)
   -order:              Ordering of the reference tensor (x, y, z) with respect to
                        ORCA output (min, mid, max)
   -plotA:              plot the HFC-tensors
   -plotQ:              plot the NQC-tensors
   -detail:             print detailed information
```

NOTE:

- By default the D-tensor is used as reference tensor only if $S > \frac{1}{2}$ and if D>0.3 cm^{-1}. In all other cases, the g-tensor is used as reference tensor. The user can manually select the reference tensor (if available in the `.property.txt` file) by using –`refg` or –`refD`.
- By default, the Euler rotation is performed using the z–y–z convention. The z-x-z convention can be selected manually by using the option –`conv zxz`.
- By default, the axes of the g- or D-tensor are assigned based on the magnitude of their components: $g_{min} \rightarrow g_x$, $g_{mid} \rightarrow g_y$, $g_{max} \rightarrow g_z$ (similarly for D). This ordering can be modified manually when running the standalone program, using the –`order` option as shown below:

-order 3 2 1:	min → z
	mid → y
	max → x
-order 1 -2 3:	min → x
	mid → y (flipped in the orientation)
	max → z

- Files in `.xyz` format can be generated to plot nuclear hyperfine and quadrupole coupling tensors using the `orca_euler` program with the option –`plotA` or –`plotQ`. For example, the HFC tensor of atom 3 (counting starts at zero) is saved as `prop-file.3.A.xyz`, and the corresponding NQC tensor as `prop-file.`

3.Q.xyz. These xyz files include four fictitious atoms (He, Ne, Ar, Kr), where the directions of the tensor axes are defined as vectors: x-axis from He to Ne; y-axis from He to Ar; and z-axis from He to Kr.

- To print the actual definition of the rotation matrix and further information on the relative orientation, use the –detail option.

9.2.12 orca_exportbasis

The orca_exportbasis utility program prints out the basis sets used by ORCA. It requires the name of the basis set as specified in the ORCA input line. Optional arguments include an ECP basis set, a list of specific atoms, and the name of an output file. The output is written in ASCII format, which can be viewed and manually edited. Users can choose to export the basis sets in either ORCA format, which can be copied directlyinto the input file, or in GAMESS-US format, which can be used via the %basis block as externally specified basis.

NOTE: Basis set names containing special characters may require a pair of quotation marks (" or ') to be correctly recognized.

The usage and options of orca_exportbasis are as follows:

```
USAGE: orca_exportbasis keywords options

       -b, --basis  : name of basis set
                      def2-svp
                      'def2-tzvp(-f)' - string to be passed with literals

EXAMPLE: orca_exportbasis -b svp
```

```
Additional Options:

       -e, --ecp    : ecp basis
                      sdd
                      default        - ECP-part of basis (if present)

       -f, --format : output format
                      ORCA           - to be read via %basis NewGTO
                      GAMESS-US      - to be read as %basis GTOName 'mybasis.bas'
                      default        - ORCA
       -a, --atoms  : list of elements
                      Cu             - single element
                      Ga Ge As Se    - list of elements separated by blanks
                      default        - whole periodic table is printed

       -o, --outfile: name of outputfile
                      mybasis.bas
                      default        - derived name

EXAMPLE: orca_exportbasis -b svp -e sdd -a Ag -f GAMESS-US -o mybasis.bas
```

The output file stored in GAMESS-US format can be called in the %basis block of the next ORCA calculation - see *Reading Basis Sets from a File*.

```
%basis
GTOName      "mybasis.bas"
GTOAuxJName  "myauxjbasis.bas"
GTOAuxJKName "myauxjkbasis.bas"
GTOAuxCName  "myauxjcbasis.bas"
end
```

9.2.13 orca_eca

The `orca_eca` utility program uses the calculated exchange coupling constants to compute relative energies of all possible spin states by diagonalizing the spin Hamiltonian. The absolute and relative energies of the resulting spin states are printed in the `*.en` and `*.en0` files, respectively. In addition, the on-site spin expectation values are written to a `*.sp` file. The following example calculates the spin ladder for a system with exchange coupling constant of -152.48 cm**-1 between Mn(III) and Mn(IV).

```
%sim
ms_bs 0.5 # Arbitrary spin state
end

# specification of spin centers
$spins 2
1 2.0  # Spin on first manganese
2 1.5  # Spin on second manganese

# Exchange coupling constant (H = -2J S1 S2)
$ecc 1
1 -152.48

$aiso_bs 2 # A false segment just to print the *.sp file
1 0.00
2 0.00
```

9.2.14 orca_pnmr

The `orca_pnmr` program calculates the paramagnetic contribution to the NMR shielding tensor using the EPR g, A, and D tensors (see Section *Paramagnetic NMR Shielding Tensors* for theoretical background). It is a standalone program that can be executed from the command line after the main ORCA calculation has finished. Alternatively, it can read user-provided $g/A/D$ tensors from an input file using the option `-i`. Note that `orca_pnmr` expects g and A tensors to follow the conventions described in Section *Cartesian Index Conventions for EPR and NMR Tensors*.

The usage and options of `orca_pnmr` are as follows:

```
USAGE:    orca_pnmr BaseName [-i] [-v]

OPTIONS:  -i  : read from the input file "BaseName.pnmr.inp"
          -v  : print more output (Z matrices)
```

When run without any options, `orca_pnmr` attempts to extract the g, D, and A tensors from the file `BaseName.property.txt` and compute the paramagnetic shieldings at 298 K. This functionality only works if the tensors were computed using the EPRNMR module. A more flexible and controlled approach is to manually edit the `BaseName.pnmr.inp` (see below for its content) and then run `orca_pnmr` with the option `-i`.

```
3    # Spin multiplicity (2S+1)
298  # Temperature range minimum [K]
300  # Temperature range maximum [K]
1    # Temperature step [K]
1    # Have g-tensor? 0 or 1
  2.004689   0.000000  -0.000000 # Cartesian g-tensor
  0.000000   2.004689   0.000000 # (if available)
  0.000000   0.000000   2.002123
1    # Have D-tensor? 0 or 1
 -0.514341  -0.000000  -0.000000 # Cartesian D-tensor [cm-1]
 -0.000000  -0.514341  -0.000000 # (if available)
 -0.000000  -0.000000   1.028682
2    # Number of A-tensors
0O   # First nucleus (index,element)
-72.358765                      # Prefactor [MHz]
```

(continues on next page)

(continued from previous page)

```
-78.989514   0.000000  -0.000000 # Cartesian A-tensor [MHz]
  0.000000 -78.989514  -0.000000
 -0.000000  -0.000000  53.478581
1O   # Next nucleus (index,element)
-72.358765                       # Prefactor [MHz]
-78.989516   0.000000  -0.000000 # Cartesian A-tensor [MHz]
  0.000000 -78.989516  -0.000000
 -0.000000  -0.000000  53.478580
# ... Further nuclei
```

- The D and g tensors are optional. If they are not provided, `orca_pnmr` assumes an isotropic free-electron g value and sets $D = 0$.
- The A tensors are required. If no A tensor is provided for a given nucleus, the paramagnetic NMR shielding cannot be computed for that nucleus.

9.2.15 orca_lft

Starting from ORCA 5.0, ORCA features a standalone multiplet program called `orca_lft`.

- `orca_lft` is dedicated to experimental spectroscopists.
- It is able to run an arbitrary number of spectra simulations with emphasis on X-ray spectroscopies.

In this section we briefly review the main functionalities of `orca_lft`. For a more detail description and examples discussion, please refer to the `orca_lft` tutorial.

1. The goal is to be able to compute various spectroscopic properties of a given LFT center (ion) if one can manually pass the information of 1 and 2 electron integrals in the form of e.g the diagonal elements of the LFT matrix (LFT orbital energies) and the Slater-Condon parameters of a given LFT problem.
2. This will allow the experimental spectroscopist to perform a massive amount of spectra simulations during the actual running experiments

Any LFT problem can be parametrized in terms 1-electron H_{LF} matrix elements and the Slater-Condon 2-electron integrals F0, F2, F4, (or the Racah parameters A, B, C, of the d-shell). (Figure: Fig. 9.9)

$$\hat{H}_{LF} = \overbrace{\sum_i \hat{v}_{LF}(i)}^{1-electron\,integrals} + \overbrace{\sum_{i<j} \hat{G}_{LF}(i,j)}^{2-electron\,integrals}$$

Fig. 9.9: Definition of an LFT problem in terms of 1-electron energies and Slater Condon parameters (SCPs)

In practice we need to know:

1. the Slater Condon parameters of a given LFT problem
2. the H_{LF} matrix elements or the relation of them (ligand field splitting, 10Dq, AOM model)

The design workflow of orca_lft is the following:

- Solve the General CI problem on a User-specified LFT problem: Type of Ion, Number of Electrons, Involved Shells, Involved Multiplicities
- Compute All possible Non Relativistic States/Multiplicity
- Compute the Transition Densities on the CSFs basis
- Compute the Needed transition Moments in the given LFT basis (i.e. 2p3d)

- Compute various properties with emphasis to X-ray spectroscopy (ABS, XAS, XES, RIXS) at the Non Relativistic Limit
- Compute the respective Relativistically corrected States on the Quasi Degenerate Perturbation Theory basis
- Provide access to various relativistically corrected properies (ABS, XAS, XES, RIXS, MCD, XMCD, GTensors, Dtensors, Hyperfine Couplings, Electric Field Gradients)

`orca_lft` requires its own input. By simply executing it from the terminal

```
orca_lft
```

one gets printings for the usage:

```
*****************************************************************************************
Simulate or Fit Spectra
*****************************************************************************************

=========================================================================================
Usage: orca_lft BaseName.lft.inp [options]
=========================================================================================

-----------------------------------------------------------------------------------
 ↪-------------------------
[Options]:
-----------------------------------------------------------------------------------
 ↪-------------------------
-sim                          Simulate Spectra
-fit                          Fit Spectra (This is not yet availiable!)
-----------------------------------------------------------------------------------
 ↪-------------------------

*****************************************************************************************
Generate Initial Input
*****************************************************************************************

=========================================================================================
Usage: orca_lft BaseName [options]
=========================================================================================
```

various Run Options:

```
-----------------------------------------------------------------------------------
 ↪-------------------------
[Options]:
-----------------------------------------------------------------------------------
 ↪-------------------------
-p_case                       Requests  p-shell case
-d_case                       Requests  d-shell case
-f_case                       Requests  f-shell case
-sp_case                      Requests sp-shell case
-ps_case                      Requests sp-shell case
-sd_case                      Requests sd-shell case
-ds_case                      Requests ds-shell case
-sf_case                      Requests sf-shell case
-fs_case                      Requests fs-shell case
-pd_case                      Requests pd-shell case
-dp_case                      Requests dp-shell case
-pf_case                      Requests pf-shell case
-fp_case                      Requests fp-shell case
-df_case                      Requests pf-shell case
-fd_case                      Requests fp-shell case
-spd_case                     Requests spd-shell case
```

(continues on next page)

(continued from previous page)

```
-spf_case                      Requests spf-shell case
-sdf_case                      Requests sdf-shell case
-pdf_case                      Requests pdf-shell case
-soc                           Requests Input with SOC Constants
-atnoN                         Sets Atomic Number N
-------------------------------------------------------------------------------
↪-------------------------
-Special Cases for known Elements. LFT Parameters are filled from an internal␣
↪AILFT NEVPT2 database

-------------------------------------Supported Oxidation States-------------------
↪-------------------------
Presently Default Oxidation States are supported except for Fe:
Main Elements                  -> O
TM Elements (Fe(26))           -> I,II,III
TM Elements (All other)        -> II
Lanthanide/Actinide Elements   -> III
--------------------------------------------Valence Cases--------------------------
↪-------------------------
-atnoN  -2s2p_case             Requests 2s2p-shell case for element with atomic␣
↪number N (4-9)
-atnoN  -3s3p_case             Requests 3s3p-shell case for element with atomic␣
↪number N (12-18)
-atnoN  -4s4p_case             Requests 4s4p-shell case for element with atomic␣
↪number N (20,31-36)
-atnoN  -4s4p_case             Requests 5s5p-shell case for element with atomic␣
↪number N (38,49-54)
-atnoN  -3d4s_case             Requests 3d4s-shell case for element with atomic␣
↪number N (22-29)
-atnoN  -4d5s_case             Requests 4d5s-shell case for element with atomic␣
↪number N (40-47)
-atnoN  -5d6s_case             Requests 4d6s-shell case for element with atomic␣
↪number N (72-79)
-atnoN  -4f5d_case             Requests 4d5d-shell case for element with atomic␣
↪number N (59-70)
-atnoN  -5f6d_case             Requests 5d6d-shell case for element with atomic␣
↪number N (91-102)
----------------------------------------Core-Valence Cases------------------------
↪-----------------------
-atnoN  -1s3d_case             Requests 1s3d-shell case for element with atomic␣
↪number N (22-29)
-atnoN  -2p3d_case             Requests 2p3d-shell case for element with atomic␣
↪number N (22-29)
-atnoN  -3p3d_case             Requests 3p3d-shell case for element with atomic␣
↪number N (22-29)
-------------------------------------Core-Valence XES Cases-----------------------
↪-------------------------
-atnoN  -1s2p3d_case           Requests 1s2p3d-shell case for element with atomic␣
↪number N (22-29)
-atnoN  -1s3p3d_case           Requests 1s3p3d-shell case for element with atomic␣
↪number N (22-29)
-------------------------------------------------------------------------------
↪-------------------------
```

and various Spectra Simulation Options:

```
*******************************************************************************
Spectra Simulation Options for the BaseName.lft.inp:
*******************************************************************************


-------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
↪-------------------------
TIP: Switch ON a Property calculation as DoProperty true:
-------------------------------------------------------------------------------
↪-------------------------

-------------------------------------------------------------------------------
↪-------------------------
General Input Parameters:
-------------------------------------------------------------------------------
↪-------------------------
NEl                             Sets the number of electrons
Shell_PQN                       Sets the principle quantum number per type of shells␣
↪(s,p,d,f)
LFTCase                         !!!ALTERNATIVE TO Shell_PQN!!! Sets the given LFT␣
↪problem (2p3d, 1s3p3d, ...)
-------------------------------------------------------------------------------
↪------------------------
LFTCase WILL replace Shell_PQN
-------------------------------------------------------------------------------
↪------------------------
(e.g. Shell_PQN = 0,2,3,0 for a 2p3d calculation)
Mult                            Sets the Multiplicity/Multiplicities
NRoots                          Sets the number of Roots/Multiplicity
TMultiplets (0.01)              Threshold for the Multiplets grouping in eV
DoeV                            All values in eV. This is default. If set false the␣
↪cm-1 unit is used throughout
DoRAS                           Requests a RASCI calculation
RAS(nel: m1 h/ m2 / m3 p)       Computes the X-Ray Emission Spectra

RAS-reference with nel electrons
m1= number orbitals in RAS-1
h = max. number of holes in RAS-1
m2= number of orbitals in RAS-2 (any number of electrons or holes)
m3= number of orbitals in RAS-3
p = max. number of particles in RAS-3

DoElastic                       Computes in addition the Elastic Scattering terms in␣
↪RIXS/RIXSSOC calculations
-------------------------------------------------------------------------------
↪-------------------------
Non Relativistic Spectroscopic Properties:
-------------------------------------------------------------------------------
↪-------------------------
DoABS/DoXAS                     Computes the Absorption like Spectra
DoCD                            Computes the CD Spectra
DoXES                           Computes the X-Ray Emission Spectra
DoRIXS                          Computes the RIXS Spectra
DoQuadrupole                    Computes the ABS,XAS,RIXS Spectra beyond dipole␣
↪approximation
-------------------------------------------------------------------------------
↪-------------------------
Relativistically Corrected Spectroscopic Properties:
-------------------------------------------------------------------------------
↪-------------------------
DoSOC                           Requests the Spin Orbit Coupling Calculations
Note that this turned on automatically if zeta SOC
constant are provided
-------------------------------------------------------------------------------
↪-------------------------
DoABS/DoXAS                     Computes the SOC Corrected Absorption like Spectra
DoCD                            Computes the SOC Corrected CD Spectra
```

(continues on next page)

(continued from previous page)

```
DoMCD/DoXMCD                  Computes the SOC Corrected MCD/XMCD Spectra
DoXESSOC:                     Computes the SOC Corrected XES Spectra
DoRIXSSOC:                    Computes the SOC Corrected RIXS Spectra
DoQuadrupole                  Computes the SOC Corrected ABS,XAS, XES RIXS Spectra␣
↪beyond dipole approximation
--------------------------------------------------------------------------------
↪-------------------------
Magnetic Properties:
--------------------------------------------------------------------------------
↪-------------------------
DoMagnetization               Computes the Magnetization
DoSusceptibility              Computes the Susceptibilities
DoGTensor                     Computes the g-Tensors/Matrices
DoDTensor                     Computes the Zero-Field Splittings
DoATensor                     Computes the Hyperfine Tensors
DoEFGTensor                   Computes the Electric Field Gradient Tensors
(also Moesbauer Parameters in the presence of Fe centers)
--------------------------------------------------------------------------------
↪-------------------------
Variable Parameters (if they are not given, default values are used)
--------------------------------------------------------------------------------
↪-------------------------
Temperature(300)              Temperature to be used in the SOC calculations
MagneticField(0)              Magnetic Field (in Gauss)
NPointsPsi(10)                Solid Angle Integration points Psi for MCD and XMCD
NPointsPhi(10)                Solid Angle Integration points Phi for MCD and XMCD
NPointsTheta(10)              Solid Angle Integration points Theta for MCD and XMCD
--------------------------------------------------------------------------------
↪-------------------------
Variable Parameters needed for Magnetization and Susceptibility calculations
--------------------------------------------------------------------------------
↪-------------------------
LebedevIntegrationPoints(26)  Number for Integration points for Lebedev)
LebedevPrec(5)                Precision of the grid for different field directions
(meaningful values range from 1 (smallest) to 10 (largest)
nPointsFStep (5)              Number of steps for numerical differentiation
(def: 5, meaningful values are 3, 5 7 and 9)
MAGTemperatureMIN(4.0)        Minimum Temperature (K) for Magnetization
MAGTemperatureMAX(4.0)        Maximum Temperature (K) for Magnetization
MAGTemperatureNPoints(1)      Number of Temperature points for Magnetization
MAGFieldStep(100.0)           Size of Field step for numerical differentiation␣
↪(def: 100 Gauss)
MAGFieldMin(0.0)              Minimum Field (Gauss) for Magnetization
MAGFieldMax(70000.0)          Minimum Field (Gauss) for Magnetization
MAGNPoints(15)                Number of Field points for Magnetization
SUSTempMin(1)                 Minimum Temperature (K) for Susceptibility
SUSTempMxn(300.0)             Maximum Temperature (K) for Susceptibility
SUSNPoints(300)               Number of Temperature points for Susceptibility
SUSStatFieldMIN(0.0)          Minimum Static Field (Gauss) for Susceptibility
SUSStatFieldMAX(1)            Maximum Static field (Gauss) for Susceptibility
SUSStatFieldNPoints(1)        Number of Static Fields for Susceptibility
--------------------------------------------------------------------------------
↪-------------------------
```

`orca_lft` can run standalone by processing an input file (`Basename.lft.inp`) with orca_lft

```
orca_lft BaseName.lft.inp -sim
```

Alternatively, one can call the main orca program like

```
orca BaseName.lft.inp
```

The different benefits of the two runs are provided in the `orca_lft` tutorial

`orca_lft` can also be used to automatically generate initial proper input files. For example, an initial bare input file `basename.lft_pd.inp` can be generated by running

```
orca_lft BaseName -pd_case
```

```
-----------------------------------------------------------------
Initial Input: BaseName.lft_pd.inp for Orca_LFT has been generated:
-----------------------------------------------------------------
```

The generated example input must be filled in with the required LFT parameters and the desired spectroscopic properties, after which the simulation can be started—just like in any standard multiplet program.

```
%lft


#-----Parameters------
NEl= 0
Shell_PQN= 0, 2, 3, 0
Mult= 0
NRoots= -1
#--------------------


#---Slater-Condon Parameters---
#---All Values in eV---
PARAMETERS
F0pp =  0.00
F2pp =  0.00
F0dd =  0.00
F2dd =  0.00
F4dd =  0.00
F0pd =  0.00
F2pd =  0.00
G1pd =  0.00
G3pd =  0.00
end
#--------------------


#---Diagonal LFT-Matrix Elelemnts---
#---All Values in eV---
FUNCTIONS
0   0 " 0.00"
1   1 " 0.00"
2   2 " 0.00"
3   3 " 0.00"
4   4 " 0.00"
5   5 " 0.00"
6   6 " 0.00"
7   7 " 0.00"
end
#--------------------


#---SPECTRA/PROPERTIES---
DoABS true
#---------
end
```

(continues on next page)

(continued from previous page)

```
*xyz 0 0
Atom 0.00 0.00 0.00
*
```

Special initial inputs based on an internal NEVPT2 database can also be generated. For the 2p3d LFT case of Ni(II), this can be done as follows:

```
orca_lft BaseName -atno28 -2p3d_case
```

```
-------------------------------------------------------------------------------
Creating input for Atom Ni(II) ...
-------------------------------------------------------------------------------

-----------------------------------------------------------------
Initial Input: BaseName.lft_pd.inp for Orca_LFT has been generated:
-----------------------------------------------------------------
```

The generated `BaseName.lft_pd.inp` file includes pre-filled LFT parameters from an internal NEVPT2 database, based on precomputed CASCI/NEVPT2 AILFT values. The resulting input-ready to runfor Ni^{II}-is as follows:

```
%lft


#-----Parameters------
NEl= 14
LFTCase 2p3d
Mult= 3, 1
NRoots= 25, 30
#--------------------


#---Slater-Condon Parameters---
#---All Values in eV---
PARAMETERS
F0pp =  85.88
F2pp =  54.77
F0dd =  23.31
F2dd =  13.89
F4dd =  9.14
F0pd =  33.03
F2pd =  7.76
G1pd =  6.42
G3pd =  2.11
end
#--------------------


#---Diagonal LFT-Matrix Elelemnts---
#---All Values in eV---
FUNCTIONS
0   0 " 0.00"
1   1 " 0.00"
2   2 " 0.00"
3   3 "1138.35"
4   4 "1138.35"
5   5 "1138.35"
6   6 "1138.35"
7   7 "1138.35"
end
#--------------------
```

(continues on next page)

(continued from previous page)

```
#---SPECTRA/PROPERTIES---
DoABS true
#---------
end


*xyz 2 3
Ni 0.00 0.00 0.00
*
```

Alternatively, as discussed in the Ab initio Ligand Field Theory section (*1- and 2-shell Abinitio Ligand Field Theory (AILFT)*), one may actually run a 2-shell AILFT calculation and produce the respective `.nevpt2.lft.inp` file

```
!NoIter NEVPT2 def2-SVP def2-SVP/C


%method
frozencore fc_none
end

#--------------------
#Rotate Orbitals
#--------------------
%scf
rotate
{2,6,90}
{3,7,90}
{4,8,90}
end
end

#--------------------
#General Options
#--------------------
%casscf
nel 14
norb 8
mult 3,1
nroots 100,100

LFTCase 2p3d
rel
dosoc true
end
end


*xyz 2 3
Ni      0.0000000000      0.0000000000      0.0000000000
*
```

The structure of an `orca_lft` input is the following:

It contains:

- The General Parameters Block where the LFT problem is defined

```
-----Parameters------
NEl= 14
LFTCase 2p3d
```

(continues on next page)

(continued from previous page)

```
#Shells_PQN 0,2,3,0, Alternative definition using s,p,d,f main quantum numbers␣
↪Mult= 3, 1
NRoots= 25, 30
--------------------
```

- The PARAMETERS Block where the SCPs and SOC constant parameters are defined

```
---Slater-Condon Parameters---
---All Values in eV---
PARAMETERS
F0pp = 85.88
F2pp = 54.77
F0dd = 23.31
F2dd = 13.89
F4dd = 9.14
F0pd = 33.03
F2pd = 7.76
G1pd = 6.42
G3pd = 2.11
end
--------------------
```

```
---SOC-CONSTANTS---
---All Values in eV---
PARAMETERS
ZETA_P = 10.68
ZETA_D = 0.08
end
--------------------
```

- The FUNCTIONS Block where the LFT matrix is defined

```
---Diagonal LFT-Matrix Elelemnts---
---All Values in eV---
FUNCTIONS
0   0 " 0.00"
1   1 " 0.00"
2   2 " 0.00"
3   3 "1138.35"
4   4 "1138.35"
5   5 "1138.35"
6   6 "1138.35"
7   7 "1138.35"
end
--------------------
```

- The Properties Block where the desire simulation properties are specified

```
---SPECTRA/PROPERTIES---
DoABS true
--------------------
```

- The xyz Block where the ion, charge, multiplicity and coordinates (0. 0. 0.) are defined

```
*xyz 2 3 Ni 0.0000000000 0.0000000000 0.0000000000
```

Note

It should be emphasized the `orca_lft` through the FUNCTIONS and PARAMETERS subblocks provides

- Arbitrary parameterization of both the one-electron LFT matrix and the Slater–Condon parameters
- Powerful parameter scan capabilities

which helps to performs any kind of simulation without any symmetry restrictions. An example regarding the use of the FUNCTIONS and PARAMETERS subblocks is provided below, while a detailed `orca_lft` tutorial with worked out examples will be soon become available.

Let us perform the Ni^{2+} L-edge XAS spectrum simulation using the following input:

```
%lft


  #-----Parameters------
  NEl= 14
  LFTCase 2p3d
  Mult= 3, 1
  NRoots= 25, 30
  #--------------------


  #---Slater-Condon Parameters---
  #---All Values in eV---
  PARAMETERS
    F0pp =  85.88
    F2pp =  54.77
    F0dd =  23.31
    F2dd =  13.89
    F4dd =  9.14
    F0pd =  33.03
    F2pd =  7.76
    G1pd =  6.42
    G3pd =  2.11
  end
  #--------------------


  #---Diagonal LFT-Matrix Elelemnts---
  #---All Values in eV---
  FUNCTIONS
    0   0 " 0.00"
    1   1 " 0.00"
    2   2 " 0.00"
    3   3 "1138.35"
    4   4 "1138.35"
    5   5 "1138.35"
    6   6 "1138.35"
    7   7 "1138.35"
  end
  #--------------------

  #---SOC-CONSTANTS---
  #---All Values in eV---
  PARAMETERS
    ZETA_P = 11.341
    ZETA_D = 0.085
  end
  #--------------------

  #---SPECTRA/PROPERTIES---
  DoABS true
  Rel
```

(continues on next page)

(continued from previous page)

```
    DoSOC true
  End
  #---------
end


*xyz 2 3
Ni 0.00 0.00 0.00
*
```

Note

- From ORCA 6.1 orca_lft is properly connected with the QDPT driver
- This implies that all QDPT properties *Magnetic Properties Through Quasi Degenerate Perturbation Theory* can be requested within the Rel block

In the first step, the LFT problem at hand is defined:

```
-----------------------------------------------------------------------------
                      L I G A N D    F I E L D    T H E O R Y
-----------------------------------------------------------------------------


=============================================================================
                      LFT PARAMETERS DEFINITION BLOCK
=============================================================================


Number of electrons        = 14
Multiplicities             = 3 1
Roots                      = 25 30
Shells included            = 0 2 3 0


-------------------------------------------------------
Definition of the ligand field basis set:
-------------------------------------------------------
   0  = pz
   1  = px
   2  = py
   3  = dz2
   4  = dxz
   5  = dyz
   6  = dx2y2
   7  = dxy


-------------------------------------------------------
Definition of the static ligand field by the user:
There are 11 ligand field parameters

-------------------------------------------------------
  Nr.                Name       Initial Value
  ----------------------------------------
    1                F0PP          85.880000
    2                F2PP          54.770000
    3                F0DD          23.310000
    4                F2DD          13.890000
```

(continues on next page)

(continued from previous page)

```
    5                F4DD           9.140000
    6                F0PD          33.030000
    7                F2PD           7.760000
    8                G1PD           6.420000
    9                G3PD           2.110000
   10              ZETA_P          11.341000
   11              ZETA_D           0.085000


-----------------------------------------------------------
Definition of the ligand field functions by the user:
There are 8 ligand field functions

-----------------------------------------------------------
  Nr.      H-element         value   function
  ---------------------------------------------
 1   H(0,0)       0.000000000   0.00
 2   H(1,1)       0.000000000   0.00
 3   H(2,2)       0.000000000   0.00
 4   H(3,3)    1138.350000000  1138.35
 5   H(4,4)    1138.350000000  1138.35
 6   H(5,5)    1138.350000000  1138.35
 7   H(6,6)    1138.350000000  1138.35
 8   H(7,7)    1138.350000000  1138.35
```

After that, the CI problem is defined:

```
=> Defining the CI spaces and setting up the CI              ...
================================================================================
                              CI SETUP AND SOLUTION BLOCK
================================================================================


Making Checks...
   Multiplicty =   3, #(configurations) =     28 #(CSF's) =     28 #(Roots) =    ␣
↪25
   NRoots<NCSFs Adjussting  ==> (CSF's) =     25
Setting up CI...
   Multiplicty =   3, #(configurations) =     28 #(CSF's) =     25 #(Roots) =    ␣
↪25
================================================================================
                              CI SETUP AND SOLUTION BLOCK
================================================================================


Making Checks...
   Multiplicty =   1, #(configurations) =     36 #(CSF's) =     36 #(Roots) =    ␣
↪30
   NRoots<NCSFs Adjussting  ==> (CSF's) =     30
Setting up CI...
   Multiplicty =   1, #(configurations) =     36 #(CSF's) =     30 #(Roots) =    ␣
↪30
CI setup done
```

Then, the CI problem is solved:

```
------------------------------
 LFT-CI TRANSITION ENERGIES
------------------------------
```

(continues on next page)

(continued from previous page)

```
LOWEST ROOT (ROOT 0, MULT 3) =    460.113216547 Eh 12520.317 eV

STATE ROOT MULT  DE/a.u.     DE/eV    DE/cm**-1
   1:   1    3   0.000000     0.000      0.0
   2:   2    3   0.000000     0.000      0.0
   3:   3    3   0.000000     0.000      0.0
   4:   5    3   0.000000     0.000      0.0
   5:   4    3   0.000000     0.000      0.0
   6:   6    3   0.000000     0.000      0.0
   7:   0    1   0.086266     2.347  18933.2
   8:   1    1   0.086266     2.347  18933.2
   9:   2    1   0.086266     2.347  18933.2
  10:   3    1   0.086266     2.347  18933.2
  11:   4    1   0.086266     2.347  18933.2
  12:   7    3   0.099001     2.694  21728.2
  13:   8    3   0.099001     2.694  21728.2
  14:   9    3   0.099001     2.694  21728.2
  15:   5    1   0.132624     3.609  29107.7
  16:   6    1   0.132624     3.609  29107.7
  17:   7    1   0.132624     3.609  29107.7
  18:   8    1   0.132624     3.609  29107.7
  19:   9    1   0.132624     3.609  29107.7
  20:  10    1   0.132624     3.609  29107.7
  21:  12    1   0.132624     3.609  29107.7
  22:  11    1   0.132624     3.609  29107.7
  23:  13    1   0.132624     3.609  29107.7
  24:  14    1   0.331652     9.025  72789.3
  25:  15    1  29.897078   813.541 6561650.2
  26:  16    1  29.897078   813.541 6561650.2
  27:  17    1  29.897078   813.541 6561650.2
  28:  18    1  29.897078   813.541 6561650.2
  29:  19    1  29.897078   813.541 6561650.2
  30:  10    3  29.915627   814.046 6565721.2
  31:  11    3  29.915627   814.046 6565721.2
  32:  13    3  29.915627   814.046 6565721.2
  33:  12    3  29.915627   814.046 6565721.2
  34:  14    3  29.915627   814.046 6565721.2
  35:  15    3  29.915627   814.046 6565721.2
  36:  16    3  29.915627   814.046 6565721.2
  37:  17    3  29.978158   815.747 6579445.1
  38:  18    3  29.978158   815.747 6579445.1
  39:  19    3  29.978158   815.747 6579445.1
  40:  20    3  29.978158   815.747 6579445.1
  41:  21    3  29.978158   815.747 6579445.1
  42:  22    3  30.016020   816.777 6587754.9
  43:  23    3  30.016020   816.777 6587754.9
  44:  24    3  30.016020   816.777 6587754.9
  45:  20    1  30.087356   818.719 6603411.3
  46:  21    1  30.087356   818.719 6603411.3
  47:  22    1  30.087356   818.719 6603411.3
  48:  23    1  30.106270   819.233 6607562.6
  49:  24    1  30.106270   819.233 6607562.6
  50:  25    1  30.106270   819.233 6607562.6
  51:  26    1  30.106270   819.233 6607562.6
  52:  27    1  30.106270   819.233 6607562.6
  53:  29    1  30.106270   819.233 6607562.6
  54:  28    1  30.106270   819.233 6607562.6
```

accompanied by a multiplet analysis:

```
-----------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
Atomic calculation : Multiplet analysis (LFT)
---------------------------------------------------
11 multiplets found (Threshold = 0.01 eV)
  0 3F | E =     0.000 eV | 2p(6)3d(8)
  1 3P | E =     2.694 eV | 2p(6)3d(8)
  2 3F | E =   814.046 eV | 2p(5)3d(9)
  3 3D | E =   815.747 eV | 2p(5)3d(9)
  4 3P | E =   816.777 eV | 2p(5)3d(9)
  5 1D | E =     2.347 eV | 2p(6)3d(8)
  6 1G | E =     3.609 eV | 2p(6)3d(8)
  7 1S | E =     9.025 eV | 2p(6)3d(8)
  8 1D | E =   813.541 eV | 2p(5)3d(9)
  9 1P | E =   818.719 eV | 2p(5)3d(9)
 10 1F | E =   819.233 eV | 2p(5)3d(9)
---------------------------------------------------
```

This is followed by SOC computation in the QDPT framework:

```
*************************************
Doing QDPT with ONLY SOC!
*************************************

-----------------------------------
NONZERO SOC MATRIX ELEMENTS (cm**-1)
-----------------------------------

Bra                        Ket
<Block Root   S    Ms   | HSOC |  Block Root   S    Ms>    =  Real-part     Imaginary␣
↪part
------------------------------------------------------------------------------------
↪---
0    1  1.0  1.0              0     0  1.0  1.0            0.000           -685.571
0    3  1.0  1.0              0     2  1.0  1.0            0.000            342.781
0    4  1.0  1.0              0     2  1.0  1.0            0.000              0.134
0    4  1.0  1.0              0     3  1.0  1.0            0.000             -2.946
0    5  1.0  1.0              0     2  1.0  1.0            0.000              0.987
0    5  1.0  1.0              0     3  1.0  1.0            0.000              7.920
0    5  1.0  1.0              0     4  1.0  1.0            0.000          -1022.995
0    6  1.0  1.0              0     2  1.0  1.0            0.000             -3.379

...

  1   29  0.0  0.0              0    17  1.0 -1.0        -2850.628           3039.
↪103
1   29  0.0  0.0              0    18  1.0 -1.0       -16745.172           2885.
↪219
1   29  0.0  0.0              0    19  1.0 -1.0        -1533.311           4672.283
1   29  0.0  0.0              0    20  1.0 -1.0          105.943            -43.140
1   29  0.0  0.0              0    21  1.0 -1.0           14.076             39.149

Note: In the following the full <I|HBO+SOC|J> are printed in the CI Basis.
I,J are compound indices for |Block/Mult, Ms, Root>, where the states
are ordered first by MultBlock, then Ms and finally Root.

...
```

The corrected SOC states are then printed out:

```
The threshold for printing is 0.0010
Eigenvectors:
                         Weight      Real          Image     : Block Root     Spin  ␣
```

(continues on next page)

(continued from previous page)

```
↪Ms
 STATE   0:       0.0000
                         0.014868    -0.121920     0.001813 :     0    0    1    1
                         0.158642     0.118778     0.380175 :     0    1    1    1
                         0.022316     0.003329     0.149347 :     0    2    1    1
                         0.073524    -0.037132     0.268598 :     0    3    1    1
                         0.071062    -0.251422     0.088592 :     0    4    1    1
                         0.068532    -0.250758     0.075180 :     0    5    1    1
                         0.010310     0.057750     0.083513 :     0    6    1    1
                         0.003333    -0.030103    -0.049264 :     0    0    1    0
                         0.007037     0.072029    -0.042999 :     0    1    1    0
                         0.245464     0.484256     0.104691 :     0    2    1    0
                         0.052920    -0.229815    -0.010229 :     0    3    1    0
                         0.017953     0.132002     0.022989 :     0    4    1    0
                         0.005423    -0.072174     0.014613 :     0    5    1    0
                         0.003723     0.024970     0.055669 :     0    6    1    0
                         0.018821     0.098122     0.095882 :     0    0    1   -1
                         0.081892    -0.206481     0.198135 :     0    1    1   -1
                         0.002458    -0.042784     0.025041 :     0    2    1   -1
                         0.057211    -0.099749     0.217397 :     0    3    1   -1
                         0.039565     0.098450     0.172838 :     0    4    1   -1
                         0.043388     0.182357     0.100666 :     0    5    1   -1
                         0.001044     0.019418     0.025827 :     0    6    1   -1
 STATE   1:       0.0000
                         0.042839    -0.206750    -0.009682 :     0    0    1    1
                         0.030129    -0.131357    -0.113467 :     0    1    1    1
                         0.004821    -0.067710    -0.015378 :     0    2    1    1
                         0.096567     0.259529     0.170915 :     0    3    1    1
                         0.019022    -0.134583    -0.030166 :     0    4    1    1
                         0.029125    -0.070877     0.155246 :     0    5    1    1
                         0.024443     0.150354     0.042856 :     0    6    1    1
                         0.126000    -0.140903    -0.325802 :     0    0    1    0
                         0.066620    -0.112104    -0.232493 :     0    1    1    0
                         0.034344    -0.075363    -0.169305 :     0    2    1    0
                         0.043496    -0.012839    -0.208161 :     0    3    1    0
                         0.035068    -0.017670     0.186428 :     0    4    1    0
                         0.088777    -0.019366    -0.297325 :     0    5    1    0
                         0.032394    -0.009435    -0.179735 :     0    6    1    0
                         0.033273    -0.179716    -0.031232 :     0    0    1   -1
                         0.038086    -0.175066     0.086245 :     0    1    1   -1
                         0.058972    -0.201976     0.134823 :     0    2    1   -1
                         0.091992     0.163866    -0.255226 :     0    3    1   -1
                         0.051908    -0.117870     0.194975 :     0    4    1   -1
                         0.015287    -0.106696    -0.062470 :     0    5    1   -1
                         0.036319     0.039383    -0.186462 :     0    6    1   -1
...
STATE 104: 6683223.2237
                         0.048335    -0.215907     0.041467 :     0   17    1    1
                         0.001350    -0.015566    -0.033287 :     0   18    1    1
                         0.058642    -0.018471    -0.241456 :     0   19    1    1
                         0.014743    -0.111453    -0.048181 :     0   20    1    1
                         0.019424    -0.128940     0.052904 :     0   21    1    1
                         0.013441    -0.042408    -0.107900 :     0   22    1    1
                         0.013136     0.106586    -0.042134 :     0   23    1    1
                         0.027360    -0.066625    -0.151399 :     0   24    1    1
                         0.030687     0.028711     0.172808 :     0   17    1    0
                         0.005134    -0.016659    -0.069691 :     0   18    1    0
                         0.006811    -0.016403    -0.080880 :     0   20    1    0
                         0.100510    -0.057415    -0.311791 :     0   21    1    0
                         0.054414    -0.044209    -0.229041 :     0   23    1    0
                         0.048345    -0.215766     0.042306 :     0   17    1   -1
```

(continues on next page)

(continued from previous page)

```
                        0.001057    -0.001549     0.032472 :     0   18    1   -1
                        0.060316     0.071066     0.235085 :     0   19    1   -1
                        0.015880    -0.093356     0.084643 :     0   20    1   -1
                        0.019274    -0.138702    -0.005996 :     0   21    1   -1
                        0.013436    -0.004756     0.115817 :     0   22    1   -1
                        0.013139     0.114517     0.004987 :     0   23    1   -1
                        0.026471    -0.005403     0.162611 :     0   24    1   -1
                        0.050526     0.219718    -0.047434 :     1   20    0    0
                        0.293497    -0.532819     0.097985 :     1   21    0    0
                        0.064015     0.249661    -0.041039 :     1   22    0    0
```

Then, the SOC-corrected absorption (here XAS) spectrum is provided:

```
-----------------------------------------------------------------------------------
↪----------------------
      SOC CORRECTED ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS
-----------------------------------------------------------------------------------
↪----------------------
      Transition          Energy      Energy  Wavelength fosc(D2)       D2        |DX|␣
↪      |DY|       |DZ|
                           (eV)       (cm-1)     (nm)   (*population)  (au**2)     (au)␣
↪      (au)       (au)
-----------------------------------------------------------------------------------
↪----------------------
  1-3.0A  ->  2-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  1-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  2-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  3-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  3-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  3-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  4-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  4-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  5-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  6-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  4-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  7-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  6-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  7-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  5-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  6-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  7-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  5-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  8-3.0A    0.000000       0.0      0.0   0.000000000   0.00000   0.
```

(continues on next page)

(continued from previous page)

```
↪00000   0.00000   0.00000
  2-3.0A  ->  8-3.0A    0.000000       0.0     0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  8-3.0A    0.000000       0.0     0.0   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  ->  9-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  ->  9-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  ->  9-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  1-3.0A  -> 10-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  2-3.0A  -> 10-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.00000
  0-3.0A  -> 10-3.0A    0.171933    1386.7  7211.2   0.000000000   0.00000   0.
↪00000   0.00000   0.0

...

2-3.0A  -> 45-3.0A   808.419525 6520343.9     1.5   0.001132627   0.00052   0.
↪01199   0.00520   0.01856
1-3.0A  -> 45-3.0A   808.419525 6520343.9     1.5   0.000466250   0.00021   0.
↪00950   0.00818   0.00742
0-3.0A  -> 45-3.0A   808.419525 6520343.9     1.5   0.001659029   0.00075   0.
↪01729   0.00696   0.02018
2-3.0A  -> 46-3.0A   808.419525 6520343.9     1.5   0.001163057   0.00053   0.
↪01282   0.00222   0.01896
1-3.0A  -> 46-3.0A   808.419525 6520343.9     1.5   0.000718139   0.00033   0.
↪00902   0.01359   0.00780
0-3.0A  -> 46-3.0A   808.419525 6520343.9     1.5   0.000710800   0.00032   0.
↪01397   0.01112   0.00211
2-3.0A  -> 47-3.0A   808.419525 6520343.9     1.5   0.002893681   0.00132   0.
↪02588   0.01640   0.01943
1-3.0A  -> 47-3.0A   808.419525 6520343.9     1.5   0.001285542   0.00058   0.
↪02045   0.00869   0.00955
2-3.0A  -> 48-3.0A   808.419525 6520343.9     1.5   0.000921831   0.00042   0.
↪00885   0.01380   0.01228
```

By processing the `*.out` file as usual with orca_mapspc *orca_mapspc*, the *.dat and *.stk files are generated. The XAS spectrum plotted using these files is given in Fig. 9.10.

Fig. 9.10: orca_lft simulated Ni^{2+} L-edge XAS spectrum for the Nickel atom (red) and for the Nickel atom in the presence of an Oh field

The power of orca_lft stems from the fact that is entirely paramerizable. This flexibility is enabled through the intrinsic capabilities of the PARAMETERS and FUNCTIONS subblocks

- **Parameters Scanability** Any given parameter—for example, the `ZETA_D` spin-orbit coupling (SOC) constant in the D-shell can be:
- *Switched on/off*: `ZETA_D= 0.0`
- *Scanned over a range of parameters*: `ZETA_D= 0.0, 0.1, 11`, which spans the range 0.0-0.1 eV for 11 steps.
- **Definition of New Parameters**
- *Users may define any number or type of custom parameters within the PARAMETERS subblock*: e.g. `10Dq= 1.2`
- **Algebraic Relations in the PARAMETERS and FUNCTIONS Subblocks**
- *Algebraic relations in the PARAMETERS subblock*:

```
`DOh = 1.2`
`D1 = {-0.4*DOh}`
`D2 = {0.6*DOh}`
```

- *Algebraic relations in the FUNCTIONS subblock*:

```
`3   3 "1138.35 + D2" # => dz2`
`4   4 "1138.35 + D1" # => dxz`
```

This provides the necessary flexibility to set up arbitrarily parameterized ligand field theory (LFT) models tailored to specific experimental chemical problems.

As an example, let us revisit the nickel complex discussed above and introduce a barycentric octahedral ligand field splitting of the d-orbital shell with 10Dq=1.2 eV.

The input now reads:

```
%lft


  #-----Parameters------
  NEl= 14
  LFTCase 2p3d
  Mult= 3, 1
  NRoots= 25, 30
  #--------------------


  #---Slater-Condon Parameters---
  #---All Values in eV---
  PARAMETERS
    F0pp =  85.88
    F2pp =  54.77
    F0dd =  23.31
    F2dd =  13.89
    F4dd =  9.14
    F0pd =  33.03
    F2pd =  7.76
    G1pd =  6.42
    G3pd =  2.11
    DOh = 1.2
    D1 = {-0.4*DOh}
    D2 = {0.6*DOh}
  end
  #--------------------


  #---Diagonal LFT-Matrix Elelemnts---
  #---All Values in eV---
  FUNCTIONS
    0   0 " 0.00"
    1   1 " 0.00"
    2   2 " 0.00"
    3   3 "1138.35 + D2" # => dz2
    4   4 "1138.35 + D1" # => dxz
    5   5 "1138.35 + D1" # => dyz
    6   6 "1138.35 + D2" # => dx2-y2
    7   7 "1138.35 + D1" # => dxy
  end
  #--------------------

  #---SOC-CONSTANTS---
  #---All Values in eV---
  PARAMETERS
    ZETA_P = 11.341
    ZETA_D = 0.085
  end
  #--------------------

  #---SPECTRA/PROPERTIES---
  DoABS true
  Rel
    DoSOC true
  End
  #---------
```

(continues on next page)

(continued from previous page)

```
End

*xyz 2 3
Ni 0.00 0.00 0.00
*
```

By running the above inputone see that the new parameters `DOH`, `D1` and `D2` are succesfully defined:

```
-----------------------------------------------------------
Definition of the ligand field basis set:
-----------------------------------------------------------
   0  = pz
   1  = px
   2  = py
   3  = dz2
   4  = dxz
   5  = dyz
   6  = dx2y2
   7  = dxy


-----------------------------------------------------------
Definition of the static ligand field by the user:
There are 14 ligand field parameters

-----------------------------------------------------------
  Nr.               Name       Initial Value
  ----------------------------------------
    1               F0PP          85.880000
    2               F2PP          54.770000
    3               F0DD          23.310000
    4               F2DD          13.890000
    5               F4DD           9.140000
    6               F0PD          33.030000
    7               F2PD           7.760000
    8               G1PD           6.420000
    9               G3PD           2.110000
   10                DOH           1.200000
   11                 D1          -0.480000
   12                 D2           0.720000
   13             ZETA_P          11.341000
   14             ZETA_D           0.085000
```

And the Oh LFT splitting is take into account in the 1-electron LFT matrix as:

```
-----------------------------------------------------------
Definition of the ligand field functions by the user:
There are 8 ligand field functions

-----------------------------------------------------------
  Nr.      H-element         value   function
  ----------------------------------------------
 1   H(0,0)      0.000000000   0.00
 2   H(1,1)      0.000000000   0.00
 3   H(2,2)      0.000000000   0.00
 4   H(3,3)   1139.070000000  1138.35 + D2
 5   H(4,4)   1137.870000000  1138.35 + D1
 6   H(5,5)   1137.870000000  1138.35 + D1
 7   H(6,6)   1139.070000000  1138.35 + D2
 8   H(7,7)   1137.870000000  1138.35 + D1
```

(continues on next page)

(continued from previous page)

```
=> Defining the one-electron LFT matrix                    ... done
-------------------------------------------------------
The ligand field one electron eigenfunctions:
-------------------------------------------------------
Orbital    Energy (eV) Energy(cm-1)          pz         px         py         dz2 ␣
↪      dxz        dyz       dx2-y2       dxz
    1          0.000        0.0          1.000000   0.000000   0.000000   0.
↪000000   0.000000   0.000000   0.000000   0.000000
    2          0.000        0.0          0.000000   1.000000   0.000000   0.
↪000000   0.000000   0.000000   0.000000   0.000000
    3          0.000        0.0          0.000000   0.000000   1.000000   0.
↪000000   0.000000   0.000000   0.000000   0.000000
    4       1137.870  9177541.4          0.000000   0.000000   0.000000   0.
↪000000   1.000000   0.000000   0.000000   0.000000
    5       1137.870  9177541.4          0.000000   0.000000   0.000000   0.
↪000000   0.000000   1.000000   0.000000   0.000000
    6       1137.870  9177541.4          0.000000   0.000000   0.000000   0.
↪000000   0.000000   0.000000   0.000000   1.000000
    7       1139.070  9187220.1          0.000000   0.000000   0.000000   0.
↪000000   0.000000   0.000000   1.000000   0.000000
    8       1139.070  9187220.1          0.000000   0.000000   0.000000   1.
↪000000   0.000000   0.000000   0.000000   0.000000
```

Upon successful termination, one can process the Ni L-edge XAS spectrum and plot the resulting data, as shown in Fig. 9.10 (blue line), which indeed reproduces characteristic features of the Ni L-edge XAS spectrum of, for example, $Ni(H_2O)_6$.

Note

- A detailed tutorial will soon become available that further explores the capabilites of orca_lft

9.2.16 orca_crystalprep

ORCA features a utility program `orca_crystalprep` that can process crystallographic `.cif` files or `.xyz` supercell files and produce proper inputs for the embedded cluster calculations.

To perform an embedded cluster calculation conventionally or within the Ionic-Crystal-QMMM, one needs to define basically 3 regions:

1. The quantum cluster QC that will be treated quantum mechanically.
2. The point charge region PC that represents the solid's environment.
3. A boundary region BR or ECP that is located between the QC and PC with the main role to prevent charge communication between the QC and PC regions.

This implies that, in the first step, a SuperCell (.xyz) must be generated, with different regions separated according to the calculation design. In the second step, the charge of the system must be balanced. Finally, all these need to be combined into a proper calculation input.

This is clearly a multistep and many times multiplatform process that is

1. complicated,
2. time consuming,
3. not user friendly.

The `orca_crystalprep` utility is designed to automatically generate proper inputs for ORCA embedded cluster calculations with the aim to allow to a wide range of experienced and not experienced users the ability to setup an embedded cluster calculation with a minimal effort.

`orca_crystalprep` requires its own input. By simply executing it from the terminal:

```
orca_crystalprep
```

one gets printings for the usage

```
*****************************************************************************************
Generate initial ORCA CrystalPrep Input
*****************************************************************************************

=========================================================================================
Usage: orca_crystalprep [Basename Input] [options]
=========================================================================================

-----------------------------------------------------------------------------------
↪-------------------------
[Options]:
-----------------------------------------------------------------------------------
↪-------------------------
-geninput                      Generate Initial Input
-----------------------------------------------------------------------------------
↪-------------------------

*****************************************************************************************
Generate ORCA Embedding Cluster Inputs using the CrystalPrep Utility
*****************************************************************************************

=========================================================================================
Usage: orca_crystalprep [CrystalPrep Input]
=========================================================================================
```

and the different options:

```
-----------------------------------------------------------------------------------
↪-------------------------
[CrystalPrep Input Options]:
-----------------------------------------------------------------------------------
↪-------------------------
General Definitions
-----------------------------------------------------------------------------------
↪-------------------------
DoCIF                        true                     This will process a .cif file
DoXYZ                        true                     This will process a .xyz file
InputCIF                     "CIFFileName"            Set the name of the␣
↪CIFFileName
InputXYZ                     "XYZFileName"            Set the name of the␣
↪XYZFileName
-----------------------------------------------------------------------------------
↪-------------------------
SuperCell Construction Definitions
-----------------------------------------------------------------------------------
↪-------------------------
DoSuperCell                  true                     Flag to generate a SuperCell
SCDimension                  "axbxc"                  The Dimension of the␣
↪SuperCell (e.g. "1x1x1")
InputCIF                     "CIFFileName"            Set the name of the␣
↪CIFFileName
InputXYZ                     "XYZFileName"            Set the name of the␣
↪XYZFileName
-----------------------------------------------------------------------------------
↪-------------------------
Special Tasks on SuperCells
-----------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
↪--------------------------
DoFractional                true                    Flag to Transform Cartesian␣
↪to Fractional coordinates
DoSymmetryOperations        true                    Flag to enforce using built␣
↪in Symmetry Operations
DoHemiSphereSC              true                    Flag to request a Hemisphere␣
↪Super Cell build
DoSymetricSC                true/false              Flag to request a Symmetric␣
↪Super Cell build
                            false => Origin Build    :  0 - ra,   0 - rb,   0 -␣
↪rc  => Default for Now!
                            true  => Symmetric Build :  -ra - ra, -rb - rb, -rc -␣
↪rc
-------------------------------------------------------------------------------
↪--------------------------
Embedding Cluster Definitions
-------------------------------------------------------------------------------
↪--------------------------
DoEmbedding                 true                    Flag to generate the files␣
↪for the embedding approach
UseVolumeCriterion          true                    Volume Criterion to generate␣
↪layers
UseDistanceCriterion        true                    Distance Criterion to␣
↪generate layers
CellVolumeFraction          value                   Cell(UniCell/SuperCell)␣
↪fraction (default 1.0)
DoMolecularFragments        value                   Define Molecular Fragments␣
↪during the build
DoSimpleInput               true                    Flag to generate a␣
↪conventional Embedding Cluster input
DoICQMMMInput               true                    Flag to generate a Ionic-
↪Crystal-QMMM input
DoMCQMMMInput               true                    Flag to generate a Mol-
↪Crystal-QMMM input
WritePDB                    true                    Flag to run a Ionic-Crystal-
↪QMMM input from a PDF file
QCCharge                    Charge Number           Specify the total QC Charge
QCMult                      Multiplicity Number     Specify the total Multiplicity
-------------------------------------------------------------------------------
↪--------------------------
Special Tasks on Embedding Cluster Construction/Definition
-------------------------------------------------------------------------------
↪--------------------------
DoLayers                    true                    Request Layers Definition
1) Layers Definition. There are 2 Options:
             a) The Differnt regions are build in layers as multipoles of the␣
↪UnitCell
             b) The Differnt regions are build in layers around a predefined QC␣
↪cluster via a QCAtom List
QCLayers                    QC Layers Number        Specify the number of the QC␣
↪Layers
ECPLayers                   ECP Layers Number       Specify the number of the ECP␣
↪Layers
PCLayers                    PC Layers Number        Specify the number of the PC␣
↪Layers
HFLayers                    HF Layers Number        Specify the number of the HF␣
↪Layers
Example Input
QCLayers 1
2) Atoms Definition. This is alternative to Layers Definition (e.g. DoLayers false)
NQCAtoms                    QC Atoms  Number        Specify the number of the QC␣
```

(continues on next page)

(continued from previous page)

```
↪Atoms
NHFAtoms                      HF Atoms  Number        Specify the number of the HF␣
↪Atoms
NECPAtoms                     ECP Atoms  Number       Specify the number of the ECP␣
↪Atoms
NPCAtoms                      PC Atoms  Number        Specify the number of the PC␣
↪Atoms
QCAtoms                       QC Atoms  List          Specify the List   of the QC␣
↪Atoms (e.g. 0,1,4,10...
HFAtoms                       HF Atoms  List          Specify the List   of the HF␣
↪Atoms (e.g. 0,1,4,10...
ECPAtoms                      ECP Atoms  List         Specify the List   of the ECP␣
↪Atoms (e.g. 0,1,4,10...
PCAtoms                       PC Atoms  List          Specify the List   of the PC␣
↪Atoms (e.g. 0,1,4,10...
Example Input
NQAtoms= 4
QCAtoms 0,1,4,10
--------------------------------------------------------------------------------
↪--------------------------
Request an Explicit Atom Definion in Ionic-Crystal-QMMM
HINT: This is automatically Set to true if an Atom List (QC Atoms, ECP Atoms, ...)␣
↪is provided by the user
--------------------------------------------------------------------------------
↪--------------------------
SetQCAtoms                    true                    Set explicitely the QC Atoms
SetHFAtoms                    true                    Set explicitely the HF Atoms
SetECPAtoms                   true                    Set explicitely the ECP Atoms
SetPCAtoms                    true                    Set explicitely the PC Atoms
SetPC2Atoms                   true                    Set explicitely the PC2 Atoms
--------------------------------------------------------------------------------
↪--------------------------
Load a User Defined QC Cluster
--------------------------------------------------------------------------------
↪--------------------------
LoadQCCluster                 true                    Load the QC Cluster
InputXYZ                   "QCXYZFileName"         Set the name of the QC␣
↪XYZFileName
--------------------------------------------------------------------------------
↪--------------------------
NOTE: This ALWAYS requires to build symmetric Super Cells. Hence DoSymetricSC␣
↪turns on by default!
--------------------------------------------------------------------------------
↪--------------------------
STEPS:
      1) Build a minimal (1x1x1 or 2x2x2) and define the QC cluster that is saved␣
↪as QCXYZFileName
      2) Build the actual SuperCell (e.g. 10x10x10) by Loading the QC cluster
         and define the QC cluster that is saved as QCXYZFileName
      3) Build the Embedded Cluster around the Loaded QC employin the Layers scheme
--------------------------------------------------------------------------------
↪--------------------------
Redefine SuperCell Origin. This is for shifting the center origin to a desired␣
↪atom during the SC Construction
This Helps to automatically construct desired Cluster structures using the Layers␣
↪Definition
--------------------------------------------------------------------------------
↪--------------------------
ShiftOrigin                   true                    Request Origin Shift  to a␣
↪particular Atom
ChosenAtom                    an Atom Number          The Number of the Chosen atom␣
```

(continues on next page)

(continued from previous page)

```
↪(e.g. 2)
--------------------------------------------------------------------------------
↪--------------------------
Neutralize the Embedded Cluster in the Simple Input
Note that in the Ionic-Crustal QMMM Input this step is taken care on demand during␣
↪the QM/MM run
--------------------------------------------------------------------------------
↪--------------------------
ChargePoints                the charge points        Specify the number of the␣
↪charge points (default 1000)
ChargeStep                  the charge points        Define the step during the␣
↪iterations (default 0.01)
ChargeThres                 the threshold            Define the neutralization␣
↪threshold (default 0.01)
Neutralization Schemes:
NeutralizingScheme1         true                 ->  Neutralization is based on␣
↪All Charges
NeutralizingScheme2         true                 ->  Neutralization is based on␣
↪different charges
                                                     Equiping the ECP and PC␣
↪regions
NeutralizingScheme3         true                 ->  Neutralization is based on␣
↪PC2 Region
--------------------------------------------------------------------------------
↪--------------------------
Special Definitions for the .metainfo File in the Ionic-Crustal QMMM
This helps Setting Charge/Spin in a specific atom type
It can also aid the neutralization step of the Embedded Cluster in the Simple Input
--------------------------------------------------------------------------------
↪--------------------------
NAtomTypes                  Number of Atom Types     Specify The Atom types that␣
↪will be defined
Example Input
#-------------------------
#Atom Type Charge Spin
#-------------------------
AtomTypes 2
Co    1    2.0   1.5
Co    2    3.0   0
end
This specifies the local Spin and Charge of a Td HS Co2+ center (Type 1) and a OH␣
↪Co3+ center (Type 2)
During the Embedding cluster construction
--------------------------------------------------------------------------------
↪--------------------------
```

In the first step, the `orca_crystalprep` tool can be used to generate its own input by running:

```
orca_crystalprep crystalprep.inp -geninput
```

This will generate an initial input:

```
-----------------------------------------------------------------------------
Initial Input: crystalprep.inp for Orca_CrystalPrep has been generated. All Done!
-----------------------------------------------------------------------------
```

It looks like the following:

```
%crystalprep
```

(continues on next page)

(continued from previous page)

```
#************************
#Read CIF/XYZ
#************************
DoCIF true

#------------------------
#INPUT CIF/XYZ
#------------------------
#InputCIF "CIFName.cif"
#------------------------
#Set Special Tasks
#------------------------
#SpaceGroupNumber 200

#************************
#Generate SuperCell
#************************
DoSuperCell true
SCDimension "1x1x1"

#************************
#Setup Embedding Approach
#************************
DoEmbedding true
DoLayers true

#------------------------
#Atom Type Charge Spin
#------------------------
#NAtomTypes 2
#Co 1 2.0  1.5
#Co 2 3.0  0.0

#************************
#Generate Inputs
#************************
#DoSimpleInput true
#DoICQMMMInput true

#Neutralize true
#QCCharge 0
#QCMult 1
#------------------------
end
```

As outlined in Fig. 9.11, by specifying the names of the desired `*.cif` or `*.xyz` file along with the appropriate options, a ready-to-run embedding cluster input can be generated.

Fig. 9.11: Embedded cluster IC-QM/MM Input generation

By default, the embedded cluster structure is constructed using a layer-based approach, in which different structural layers are generated as multiples—or fractions—of the unit cell.

The unit cell fractions, expressed in terms of volume units, are specified using the following keyword:

```
CellVolumeFraction N #by default N=1
```

As an example, let us consider the NaCl case in detail . We aim to generate a complete embedding cluster setup that includes:

1. a 20x20x20 supercell, constructed from the nacl.cif file,
2. an embedding cluster with 1 QC layer and 1 ECP layer, and
3. an IC-QM/MM embedding cluster input

These can be prepared with the following ORCA input file calling `orca_crystalprep` program through the `%crystalprep` block:

```
%crystalprep


#************************
#Read CIF/XYZ
#************************
DoCIF true

#------------------------
#INPUT CIF/XYZ
#------------------------
InputCIF "nacl.cif"

#************************
#Generate SuperCell
#************************
DoSuperCell true
SCDimension "20x20x20"

#************************
#Setup Embedding Approach
```

(continues on next page)

(continued from previous page)

```
#************************
DoEmbedding true
DoLayers true

#------------------------
#Atom Type Charge Spin
#------------------------
NAtomTypes 2
Na 0  1.0  0.0
Cl 1 -1.0  0.0

#************************
#Generate Inputs
#************************
DoICQMMMInput true
QCCharge 0
QCMult 1
#------------------------
end
```

In the first step, the `orca_crystalprep` processes the nacl.cif file and creates the unit cell and the requested 20x20x20 supercell:

```
------------------------------------------------------------
Reading Information from the provided CIF file: nacl.cif
------------------------------------------------------------

------------------------Unit Cell Parameters-----------------
Hermann-Mauguin Space Group:     P1
Space Group ID:                 1
Unit Cell Symmetry:
Unit Cell Volume:             46.09
Unit Cell alpha angle:        60.00
Unit Cell beta angle:         60.00
Unit Cell gamma angle:        60.00
Unit Cell alpha length:       4.024
Unit Cell beta length:        4.024
Unit Cell gamma length:       4.024
Atom   Type   AO    x     y     z    occ
Na     0     11 0.000 0.000 0.000  1.00
Cl     1     17 0.500 0.500 0.500  1.00
Done
------------------------------------------------------------

----------Making a SuperCell with 20x20x20 dimensions-----------
Unit Cell:
0          1          2
0       4.024000   0.000000   0.000000
1       0.000000   4.024000   0.000000
2       0.000000   0.000000   4.024000

Transformation Matrix:
0          1          2
0      80.480000   0.000000   0.000000
1       0.000000  80.480000   0.000000
2       0.000000   0.000000  80.480000


--------------------------------------------------------------------
Saving xyz file: nacl4.cif_20x20x20.xyz                   ...Done
--------------------------------------------------------------------
```

Then, the costruction of the embedding cluster structure is initiated:

```
------------Making a Embedding Cluster Input ------------
------------Using the Layers Approach with: ------------
QC_Layers: 1
ECP_Layers: 1
PC_Layers: 1
--------------
Preparing Inputs ...
```

In the next step, the center of the supercell (`.xyz`) is assigned to the closest atom to the center:

```
-------------------------------
The Center of XYZ Coordinates
-------------------------------
41.246, 41.246, 41.246
-------------------------------
Closest Atom to Center
-------------------------------
Na(4630) 40.240, 40.240, 40.240
-------------------------------
Shifting Origin to closest atom
-------------------------------
40.240, 40.240, 40.240
```

After that, layers are generated automatically:

```
--------------------------------------
Saving Generated Layers XYZ Files ...
--------------------------------------
Saving Layer 0 XYZ File: nacl.cif_20x20x20.xyz_0.xyz
Saving Layer 1 XYZ File: nacl.cif_20x20x20.xyz_1.xyz
Saving Layer 2 XYZ File: nacl.cif_20x20x20.xyz_2.xyz
Saving Layer 3 XYZ File: nacl.cif_20x20x20.xyz_3.xyz
Saving Layer 4 XYZ File: nacl.cif_20x20x20.xyz_4.xyz
Saving Layer 5 XYZ File: nacl.cif_20x20x20.xyz_5.xyz
Saving Layer 6 XYZ File: nacl.cif_20x20x20.xyz_6.xyz
Saving Layer 7 XYZ File: nacl.cif_20x20x20.xyz_7.xyz
Saving Layer 8 XYZ File: nacl.cif_20x20x20.xyz_8.xyz
Saving Layer 9 XYZ File: nacl.cif_20x20x20.xyz_9.xyz

--------------------------------------
Saving Generated Layers PDB File ...
--------------------------------------
```

This is followed by the embedding cluster construction:

```
--------------------------------------
Saving Generated Cluster XYZ Files ...
--------------------------------------
Saving QC XYZ File: nacl.cif_20x20x20.xyz_QC.xyz
Saving ECP Region XYZ File: nacl.cif_20x20x20.xyz_ECP.xyz
Saving PC Region XYZ File: nacl.cif_20x20x20.xyz_PC.xyz
```

Finally, the IC-QM/MM embedding cluster is generated:

```
--------------------------------------
Saving Embedding Cluster Inputs ...
--------------------------------------
Saving ICQMMM Input: nacl.cif_20x20x20.xyz.ICQMMM.inp
Done
----------------------------------------------------------------
```

Ionic-Crystal QM/MM calculations require the generation of a simple force field, which is stored in a `*.prms` file, as described in section *Overview on ORCA's Multiscale Implementation*. The necessary information, including charge and spin, is taken from a `.metainfo` file. The `orca_crystalprep` tool allows for externally setting the charge and spin in the `.metainfo` file, as described below:

First, during processing of the .cif file, atom types are assigned to all atoms detected in the asymmetric unit.

```
Atom    Type    AO     x      y      z     occ
Na      0      11 0.000 0.000 0.000  1.00
Cl      1      17 0.500 0.500 0.500  1.00
```

In the `orca_crystalprep` input, the charge and spin of specific atom types can be assigned as follows:

```
NAtomTypes 2
Na 0  1.0  0.0
Cl 1 -1.0  0.0
```

This information is then passed in the `.metainfo` file

```
18522
# atom nr. - element - atom type - formal charge - formal spin - molecule nr
0  Na   0  1  0          1
1  Na   0  1  0          1
2  Na   0  1  0          1
...

9261  Cl   1 -1  0        1
9262  Cl   1 -1  0        1
9263  Cl   1 -1  0        1
...
```

After completing the preparatory steps, the embedded cluster is constructed as shown in Fig. 9.12.

Fig. 9.12: Generated Embedded cluster. QC: Na_4Cl_4, ECP region red dots, PC region small green and purple dots

The generated input for running IC-QM/MM embeding cluster calculation is as provided below:

```
!Ionic-Crystal-QMMM
#Include Method
```

```
%qmmm
#--------Define the Cluster---------
```

(continues on next page)

(continued from previous page)

```
ORCAFFFilename= ""
Use_QM_InfoFromPDB true
Use_QM3_InfoFromPDB true
ECPLayerECP= "SDD"
#--------Charge Convergence---------
CONV_Charges false
ENFORCETOTALCHARGE true
CHARGE_TOTAL    0
PrintLevel 4
end
#-----------------------------------

*pdbfile 0  1 nacl.cif_20x20x20.pdb
```

Staring from ORCA 6.1 the orca_crystalprep utility has been expanding its capabilities in:

- **Processing a large variety of .cif files**

In principle any type of .cif files can be processed to create a variety of supecells of any chemical system ranging from molecules to solids in a large variaty of structural complexity and space group symmetries as shown in Figure Fig. 9.13. It is noteworthy that the structure of Crystallographic Information Files (.cif) has undergone considerable evolution since their introduction by the International Union of Crystallography (IUCr). In earlier .cif files—particularly those generated prior to the standardization of the CIF Core Dictionary—header sections were often inconsistent, lacking standardized data items such as _symmetry_space_group_name_H-M, _cell_length_a, or _atom_site_label. Instead, custom or abbreviated notations were frequently used, and essential metadata (e.g., units, symmetry operations, or publication references) were often incomplete or omitted altogether.In contrast, modern .cif files, compliant with IUCr Core CIF and mmCIF dictionaries, follow a rigorously defined syntax and include extensive metadata that supports interoperability, reproducibility, and automated processing. These enhancements include explicit data block naming conventions, hierarchical loop structures for atom positions, and clearly defined symmetry operations through items like _space_group_symop_operation_xyz. Given this disparity, it is imperative that contemporary .cif parsers be designed with sufficient flexibility to accommodate structural heterogeneity while ensuring accurate extraction of intrinsic crystallographic information. This includes the correct identification and interpretation of symmetry operations and lattice parameters for all 230 crystallographic space groups and their corresponding subgroups, as defined by the International Tables for Crystallography. Such robustness is critical for enabling reliable symmetry analysis, structure refinement, and high-throughput computational workflows in crystallography and materials science. In this context Orca_crystalprep contains its own symmetry operations library and it will try to repair any potential deficiency contained in the .cif file. It is activated automatically if symmetry operations are not detected or alternatively it can be requested through the keyword `SymmetryOperations true`

For example in the case of CeO2 the orca_crystalprep input file for the generation of a spherical supercell will look like the following

Fig. 9.13: Various generated supercells by employing the orca_crystalprep utility program

```
%crystalprep


#************************
#Read CIF/XYZ
#************************
DoCIF true                #Turn on the cif file reading

#------------------------
#INPUT CIF/XYZa
#------------------------
InputCIF "CeO2.cif"       #The cif file
#------------------------
#Set Special Tasks
#------------------------

#************************
#Generate SuperCell
#************************
DoSuperCell true           #Turn on the supecell construction
SCDimension "10x10x10"     #Construct a 10x10x10 supercell
DoSymetricSC true          #Construct symmetric supercell
DoSymmetryOperations true  #Use the buildin Crystallographic database
DoHemiSphereSC true/false  #Build the supecell in cubic or hemisphere fashion
PrintLevel 2
```

(continues on next page)

(continued from previous page)

```
#************************
#Setup Embedding Approach
#************************
DoEmbedding false          #Turn off the Embended Cluster generation
```

Running this will process the .cif file

```
*************** ORCA CRYSTALPREP TOOL ************

------------------------------------------------------------
Reading Information from the provided CIF file: CeO2.cif
------------------------------------------------------------

------------------------Unit Cell Parameters-----------------
Hermann-Mauguin Space Group:     P1
Space Group ID:                 1
Unit Cell Symmetry:
Unit Cell Volume:            163.40
Unit Cell alpha angle:        90.00
Unit Cell beta angle:         90.00
Unit Cell gamma angle:        90.00
Unit Cell alpha length:       5.467
Unit Cell beta length:        5.467
Unit Cell gamma length:       5.467
Atom   Type  AO    x     y     z    occ   ox
 Ce     0    58 0.000 0.000 0.000  1.00  4.00
 Ce     1    58 0.000 0.500 0.500  1.00  4.00
 Ce     2    58 0.500 0.000 0.500  1.00  4.00
 Ce     3    58 0.500 0.500 0.000  1.00  4.00
 O      4     8 0.250 0.750 0.750  1.00 -2.00
 O      5     8 0.750 0.250 0.250  1.00 -2.00
 O      6     8 0.250 0.250 0.250  1.00 -2.00
 O      7     8 0.750 0.750 0.750  1.00 -2.00
 O      8     8 0.750 0.750 0.250  1.00 -2.00
 O      9     8 0.250 0.250 0.750  1.00 -2.00
 O     10     8 0.750 0.250 0.750  1.00 -2.00
 O     11     8 0.250 0.750 0.250  1.00 -2.00
Done
------------------------------------------------------------

----------Making a SuperCell with 10x10x10 dimensions-----------
Doing Symmetry Operations ...
Detected   1 Operations for
Hermann-Mauguin Space Group:     P1
Space Group ID:                  1
Done
Unit Cell:
                  0          1          2
      0       5.467000   0.000000   0.000000
      1       0.000000   5.467000   0.000000
      2       0.000000   0.000000   5.467000

Transformation Matrix:
                  0          1          2
      0      54.670000   0.000000   0.000000
      1       0.000000  54.670000   0.000000
      2       0.000000   0.000000  54.670000

-----------------------------
Super Cell Origin:   0,   0,   0
-----------------------------
```

(continues on next page)

(continued from previous page)

```
----------------------------
The Center of XYZ Coordinates
----------------------------
2.278, 2.278, 2.278

--------------------------------------------------------------------
Saving xyz file: CeO2.cif_10x10x10.xyz                   ...Done
--------------------------------------------------------------------


*************** ORCA CRYSTALPREP TOOL TERMINATED SUCCESFULLY ************
```

and will create the 10x10x10 spherical supecell shown in Figure Fig. 9.14 in a cubic (a) or in a hemisphere (b) construction fashion if the `DoHemiSphereSC` keyword is turned on. Note that while the cubic supercell construction scheme is appropriete for creating embedded clusters to tackle bulk chemical propblems (e.g. computation of the Optical band gaps of semiconductors, [599]). In contrast the hemisphere supercell build option is expected to be very usefull for constructing embedded clusters to tackle surface related problems (e.g. computation of adsorption energies of molecules on surfaces, [822]).

Fig. 9.14: Cubic (left) versus Hemisphere (right) generated 10x10x10 supercell of CeO2

- **Enable loading of a tailored QC.xyz file**

Staring from ORCA 6.1, it is possible to generate and load a tailored QC file directly in orca_crystalprep.

This is achieved by enabling the `DoSymetricSC` keyword. In this way all generated supercells share the same center. Hence in the case of CeO2, one can generate a 1x1x1 or 2x2x2 minimum supercell and construct/tailor the QC.xyz file (in this case Ce24O64.xyz). In following one can now re-execure the orca_crystalprep by now loading the QC file by setting `LoadQCCluster true`

```
%crystalprep


#************************
#Read CIF/XYZ
#************************
DoCIF true
```

(continues on next page)

(continued from previous page)

```
#------------------------
#INPUT CIF/XYZa
#------------------------
InputCIF "CeO2.cif"
InputQCXYZ "Ce24O64.xyz"
LoadQCCluster true
#------------------------
#Set Special Tasks
#------------------------
#SpaceGroupNumber 200

#************************
#Generate SuperCell
#************************
DoSuperCell true
SCDimension "10x10x10"
PrintLevel 2
#************************
#Setup Embedding Approach
#************************
DoEmbedding true
DoSymetricSC true
DoSymmetryOperations true
DoHemiSphereSC true
CellVolumeFraction 0.01
DoLayers true
ECPLayers 3
#------------------------
#Atom Type Charge Spin
#------------------------
NAtomTypes 12
 Ce     0    4.0  0.0
 Ce     1    4.0  0.0
 Ce     2    4.0  0.0
 Ce     3    4.0  0.0
 O      4    -2.0  0.0
 O      5    -2.0  0.0
 O      6    -2.0  0.0
 O      7    -2.0  0.0
 O      8    -2.0  0.0
 O      9    -2.0  0.0
 O     10    -2.0  0.0
 O     11    -2.0  0.0

#************************
#Generate Inputs
#************************
#------------------------
# Simple/IC-QMMM inputs
#------------------------
DoICQMMMInput true

#------------------------
#QCCharge 0
#QCMult 1
#------------------------
end
```

This will generate the embedded cluster shown in Figure Fig. 9.15 for the tailored QC=$[Ce_{24}O_{48}]^0$ quantum cluster and 3-layers of ECPs.

Note that in this case the different layers to equiop the ECPs and PCs regions are by default generated on the basis of the volume of the chosen QC cluster. In this case it is beneficial to use a small fraction of the unitcell volume defined by `CellVolumeFraction 0.01` that can be used as a criterion for the layers generation.

Fig. 9.15: CeO2 embedded cluster generated for the QC=$[Ce_{24}O_{48}]^0$ quantum cluster and 3-layers of ECPs

In a final step the orca_crystalprep generates the IC-QM/MM input file as

```
!Ionic-Crystal-QMMM
#Include Method


%qmmm
#--------Define the Cluster---------
ORCAFFFilename= "CeO2.cif_10x10x10.ORCAFF.prms"
Use_QM_InfoFromPDB true
Use_QM3_InfoFromPDB true
ECPLayerECP= "SDD"
#--------Charge Convergence---------
CONV_Charges false
ENFORCETOTALCHARGE true
CHARGE_TOTAL    0
PrintLevel 4
end
#-----------------------------------

*pdbfile 0  1 CeO2.cif_10x10x10.pdb
```

Note that the information on the QC, ECP and PC regions is read from the generated `.pdb` file.

- Further information on the IC-QM/MM and QM/MM modules in general can be found in section *Overview on ORCA's Multiscale Implementation*
- Additional details and examples related to the `orca_crystalprep` tool and the embedding approach are provided in the tutorial "Treating Solids with the Embedding Cluster approach".

9.3 orca_2json

This utility program supports the exchange of external ORCA data like geometry, orbitals and basisets but also of internal ORCA data like 1-electron and 2-electron integals with other programs.

9.3.1 Export ORCA data

The program reads information like geometries, basis sets, MOs etc. stored in the `.gbw` file or other equivalent ones as `.uno`, `.mp2nat`, `.qro` etc. and calculates integrals to export them in JSON standard output formats. For density information the `.densities` file must also be available. The program is called as a standalone via command line.

Syntax:

```
orca_2json BaseName.gbw -options
or
orca_2json BaseName.mp2nat -options
or
orca_2json BaseName.uno -options
```

The following ASCII and binary JSON-formats are available as command line options. It is possible to specify more than one format option.

```
  -json             Write ASCII JSON file (default)
  -bson             Write binary JSON file
  -ubjson           Write universal binary JSON specification file
  -msgpack          Write MessagePack file
```

In addition two more options are availble. The first of them is used to translate a basename.property.txt file to a corresponding one in JSON format (see *Property File*).

```
   -property        Translate a *.txt property file to a *.json one
```

Finally, orca_2json has the ability to create a .gbw file from a json file. For this one needs to use the '-gbw' option (see *Import JSON data into ORCA*).

```
   -gbw        Create a GBW file from a json one
```

Automatic export from ORCA job

It is also possible to run `orca_2json` on the GBW file automatically at the end of an ORCA job using the input:

```
%output
  JSONGBWFile true
end
```

This will simply run `orca_2json BaseName.gbw` with no additional options, respecting any applicable configuration file, if present (see below).

9.3.2 Configuration file

The data stored in the json file can be configured more individually. Some information like atom information, geometry and charge are always written in the outputfile (see *Basic Information*). Other data can be requested via keywords in a JSON-formatted configuration file that either exists for every ORCA output file or for all files in a directory. Without these configuration files **ALL** available data are stored except for the densities and the integrals. Because of the huge amount of data these are only available when explicitly requested in the configuration file.

You can specify a basename-dependend configuration file

```
BaseName.json.conf
```

or a global file used for all requests in a directory.

```
orca.json.conf
```

Most keywords in the configuration file can be activated or deactivated with true or false but some keywords like densities or output formats have more options and require a list of values. If an option is not specifically selected it is omitted.

Structure of the configuration file:

```
{
        "keyword": true/false,
        ...
        "keyword": true/false,
        "keyword": ["option",...."option"]
        ...
}
```

Example

```
{
  "MOCoefficients":false,
  "BasisSet":false
}
```

Using the above configuration file in the working directory, *orca_2json* will not export the molecular coefficients and the basis set information.

Example

Here is an example configuration file with most available keywords where everything is disabled except for the basis set information and the specified Integrals, all densities stored in the density file are requested and the output format should be ascii json and binary json.

```
{
        "MOCoefficients": false,
        "Basisset": true,
        "1elPropertyIntegrals": ["dipole", "quadrupole", "velocity", "printLinMom",
↪ "angular_momentum", "higherMoment", "soc"],
        "1elPropertyRelIntegrals": ["dipole", "quadrupole", "angular_momentum",
↪"soc"],
        "1elIntegrals": ["H", "S", "T", "V", "HMO"],
        "1elIntegralsRel": ["H", "S", "T", "V", "HMO"],
  "FockMatrix": ["F", "J", "K", "VXC", "Vsol"],
        "Vaux": false,
        "AuxBasisType": "AuxC",
        "FullTrafo": false,
        "OrbWin": [0,0,0,0,0,0,0,0],
        "2elIntegrals": [ "MO_IJKL", "RI_IAV", "RI_IJKL"],
        "2elNonRedIntegrals": false,
        "2elNonRedRIIntegrals": false,
```

(continues on next page)

(continued from previous page)

```
        "MullikenCharge": false,
        "LoewdinCharge": false,
        "Densities": ["all"],   <--- here you specify the names like "scfp" or
↪"scfr" or all
        "JSONFormats": ["json", "bson"]
}
```

9.3.3 Available information

Property File

Beside all information that we will see orca_2json can create, it can also translate the property file of ORCA (basename.property.txt) to a JSON file. This option gives access to all properties stored in the property file. (For more information on property file see *Property File*).

The **syntax** is:

```
orca_2json basename -property
```

Where

basename is the name of the property file **without** the extension *property.txt*.

Example

If we use the following ORCA input (with the name test.inp):

```
!HF
*xyz 0 1
  H 0.0 0.0 0.0
  H 0.0 0.0 0.8
*
```

ORCA will create and store on disk, a file named *"test.property.txt"*. The start of the file will look like this:

```
****************************************************
****************** ORCA 6.0.1 ******************
****************************************************
$Calculation_Status
   &GeometryIndex 1
   &version [&Type "String"] "6.0.1"
   &progName [&Type "String"] "LeanSCF"
   &Status [&Type "String"] "NORMAL TERMINATION"
$End
$Geometry
   &GeometryIndex 1
   &NAtoms [&Type "Integer"] 2
   &NCorelessECP [&Type "Integer"] 0
   &NGhostAtoms [&Type "Integer"] 0
   &CartesianCoordinates [&Type "Coordinates", &Dim(2,4), &Units "Bohr"]
              H      0.000000000000    0.000000000000    0.000000000000
              H      0.000000000000    0.000000000000    1.511780907137
$End
$SCF_Energy
   &GeometryIndex 1
   &totalEnergy [&Type "Double"]      -1.127112922023328 2e+00
$End
```

Then running *orca_2json* in the following way:

```
orca_2json test -property
```

ORCA will create a new file on disk, named *"test.property.json"*. The corresponding part of this file will look like this:


```
{
  "Calculation_Status": {
      "Status": "NORMAL TERMINATION",
      "progName": "LeanSCF",
      "version": "6.0.x"
  },
  "Geometries": [
    {
      "Geometry": {
        "Coordinates": {
            "Cartesians": [
                [
                    "H",
                    0.0,
                    0.0,
                    0.0
                ],
                [
                    "H",
                    0.0,
                    0.0,
                    1.511780907137
                ]
            ],
            "Type": "Cartesians",
            "Units": "a.u."
        },
        "NAtoms": 2,
        "NCorelessECP": 0,
        "NGhostAtoms": 0
      },
      "SCF_Energy": {
        "totalEnergy": -1.127112922023328 2
      }
    }
  ]
}
```


Note

The `property.json` structure has changed somewhat since ORCA version 6.0 and so have some of the property components. Converting an version 6.0 `property.txt` file to the current JSON format is partially possible but some data may be lost (with a warning).

Basic Information

Some basic information will always be written into the JSON-file as

per Atom	-	Coords
	-	ElementLabel
	-	ElementNumber
	-	Idx
	-	NuclearCharge
per molecule	-	BaseName
	-	Charge
	-	CoordinateUnits
	-	HFTyp
	-	Multiplicity
	-	PointGroup

For example using the following configuration file where we set everything to false:

```
{
  "MOCoefficients": false,
  "Basisset": false,
  "MullikenCharge": false,
  "LoewdinCharge": false,
  "JSONFormats": ["json"]
}
```

will still produce a json file which for the case of a H2 molecule should look like:

```
{
  "Molecule": {
    "Atoms": [
      {
        "Coords": [
          0.0,
          0.0,
          0.0
        ],
        "ElementLabel": "H",
        "ElementNumber": 1,
        "Idx": 0,
        "NuclearCharge": 1.0
      },
      {
        "Coords": [
          0.0,
          0.0,
          0.8
        ],
```

(continues on next page)

(continued from previous page)

```
        "ElementLabel": "H",
        "ElementNumber": 1,
        "Idx": 1,
        "NuclearCharge": 1.0
      }
    ],
    "BaseName": "test",
    "Charge": 0,
    "CoordinateUnits": "Angs",
    "HFTyp": "RHF",
    "Multiplicity": 1,
    "PointGroup": "C1"
  },
  "ORCA Header": {
    "Date": "2024-06-03 00:06:37 +0200",
    "Git": "548015a5a0",
    "Version": "                        Program Version 6.0  -  CURRENT   -\n"
  }
}
```

Densities

orca_2json can also export calculated densities in json format.

Densities	-	densities as available in the .densities file

2RDM

As of ORCA 6.1.0, it is possible to export 2RDM(s) in MO basis as calculated by the AutoCI module to JSON. See the *AutoCI keywords table* for the available options. The 2RDMs are not stored inside the `.densities` file, but in their own `.RDM2` files. The following 2RDMs are available

- `RDM2` (RHF, Γ^{pr}_{qs})
- `RDM2_aa` (UHF, $\Gamma^{p_\alpha r_\alpha}_{q_\alpha s_\alpha}$)
- `RDM2_ab` (UHF, $\Gamma^{p_\alpha r_\beta}_{q_\alpha s_\beta}$)
- `RDM2_bb` (UHF, $\Gamma^{p_\beta r_\beta}_{q_\beta s_\beta}$)

Consider the general CI 2RDM:

$\Gamma^{pr}_{qs} = \langle \Psi | E_{pq} E_{rs} | \Psi \rangle$.

For every available combination of `p` and `q`, there will be a matrix `r` x `s` in the JSON. The values of `p`,`q`,`r`,`s` range from `0` to the number of basis functions. For example, a fictional case in which there are only 2 basis functions and all elements of the 2RDM have been set to 0 will produce the following JSON.

```
{
    "Densities": {
        "RDM2": {
            "0,0": [[0.0, 0.0], [0.0, 0.0]] // p=0, q=0
            "0,1": [[0.0, 0.0], [0.0, 0.0]] // p=0, q=1
            "1,0": [[0.0, 0.0], [0.0, 0.0]] // p=1, q=0
            "1,1": [[0.0, 0.0], [0.0, 0.0]] // p=1, q=1
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Please note that the relaxed 2RDMs have been symmetrised to posses the following permutational symmetry $\Gamma^{pr}_{qs} = \Gamma^{qr}_{ps} = \Gamma^{ps}_{qr} = \Gamma^{qs}_{pr} = \Gamma^{rp}_{sq} = \Gamma^{sp}_{rq} = \Gamma^{rq}_{sp} = \Gamma^{sq}_{rp}$.

Syntax

"Densities" : [list of densities]

Where *list of densities* should be a list of strings with the expected densities.

NOTE By default densities, due to their potential size, are not exported to a json file.

NOTE An empty bracket syntax (*"Densities" : [ ]*) will cause the program to crash.

NOTE There is the string "All" available where the program will export all available densities.

Electron Integrals

The list of available electron integrals is shown in the next table.

1elIntegrals	-	1-electron integrals
1elPropertyIntegrals	-	1-electron property integrals
1elIntegralsRel	-	relativistic 1-electron integrals
1elPropertyRelIntegrals	-	relativistic 1-electron property integrals
2elIntegrals	-	two-electron integrals
2elNonRedIntegrals	-	non-redundant two-electron integrals
2elNonRedRIIntegrals	-	non-redundant two-electron RI integrals
FockMatrix	-	two-electron contributions to the Fock matrix

1-electron integrals

For 1-electron integrals we use the following notation:

H	-	one electron matrix
HMO	-	one electron matrix in MO basis
S	-	overlap matrix
T	-	kinetic energy matrix
V	-	nuclear attraction matrix

Example

```
{
  "1elIntegrals": ["H", "S"],
  "JSONFormats": ["json"]
}
```

will produce a json file were only the H-Matrix and the Overlap matrix are printed (beside the basic information). Please note that for the one electron relativistic integrals there is a separate variable (see *1-electron relativistic integrals*)

1-electron property integrals

Also available are 1-electron property integrals.

1elPropertyIntegrals	-	1-electron property integrals

Currently the following options are valid:

angular_momentum	-	Angular momentum integrals
dipole	-	Dipole moment integrals
higherMoment	-	Octupole moment integrals
quadrupole	-	Quadrupole moment integrals
velocity	-	Velocity integrals
soc	-	Spin-orbit coupling integrals

Example

```
{
  "1elPropertyIntegrals": ["dipole", "quadrupole"],
  "JSONFormats": ["json"]
}
```

Note

SOC integrals are computed with the settings chosen in the GBW file, except for the relativistic Hamiltonian (see *1-electron relativistic property integrals* for the relativistic version). In the case of SOMF, the SCF density from the density container is used.

1-electron relativistic integrals

1-electron relativistic integrals follow the same notation as the corresponding non-relativistic ones (see *1-electron integrals*).

Example

```
{
  "1elIntegralsRel":["H", "T", "V"],
  "JSONFormats": ["json"]
}
```

This example will produce and store in the corresponding json file the X2C-transformed H-Matrix, as well as the separate potential and kinetic contributions. Note that while the "relativistic" overlap matrix is available for completeness, in X2C it is identical to the non-relativistic overlap by construction.

1-electron relativistic property integrals

Also available are 1-electron relativistic (X2C) property integrals similar to the non-relativistic ones but with reduced options.

1elPropertyRelIntegrals	-	relativistic 1-electron property integrals

Currently the following options are valid:

angular_momentum	-	Angular momentum integrals
dipole	-	Dipole moment integrals
quadrupole	-	Quadrupole moment integrals
soc	-	Spin-orbit coupling integrals

Example

```
{
    "1elPropertyRelIntegrals": ["dipole","quadrupole","angular_momentum"]
}
```

Origin setting

The origin of the electric property is per default the Cartesian origin but also the center of mass and the center of nuclear charges can be selected. Additionally an arbitrary position can be given as x,y,z coordinates when ori_el = 3 is chosen. Currently the following options are valid:

ori_el	-	0 - Cartesian origin
	-	1 - center of mass
	-	2 - center of nuclear charge
	-	3 - arbitrary position
ori_el_xyz	-	position of the origin(x,y,z)

Example

```
{
    "1elPropertyIntegrals": ["dipole","quadrupole"],
    "ori_el": 3,
    "ori_el_xyz": [0.0, 1.0, 1.0]
}
```

Fock matrix contributions

The following two-electron contributions to the AO-basis Fock matrix can be exported:

J	-	Coulomb
K	-	Exchange
VXC	-	DFT XC potential
Vsol	-	Implicit solvent potential
F	-	Total Fock matrix

Note

- All terms are computed using the SCF (spin-)density from the .densities file (named "scfp" and "scfr").
- The same settings and approximations (RI, COSX, grids) as in the original calculation are used.
- All prefactors are already included, such that $\mathrm{F} = \mathrm{J} + \mathrm{K} + \mathrm{VXC} + \mathrm{Vsol}$.
- For consistency, the output dimensions are always $N_{\text{spin-ops}} \times N_{\text{AO}} \times N_{\text{AO}}$, even though all matrices are symmetric and some terms (J and Vsol) are equal for alpha/beta in the UHF case.
- The 1-electron contributions ("H") are not included and must be requested separately (see *1-electron integrals*, as well as *1-electron relativistic integrals* for the relativistic case).

Example

```
{
  "1elIntegrals": ["H"],
  "FockMatrix": ["F", "J", "K", "VXC", "Vsol"],
  "JSONFormats": ["json"]
}
```

2-electron integrals

ORCA_2json can produce and write on disk three main categories of 2-electron integrals.

1. Two-electron integrals in atomic basis (*2-electron integrals in AO basis*)
2. Two-electron integrals in molecular basis (*2-electron integrals in MO basis*)
3. Two-electron integrals using the resolution of identity approximation (RI). (*RI 2-electron integrals*)

2-electron integrals in AO basis

In atomic basis the two-electron integrals can be saved in Coulomb order or in Exchange order. The keywords for the two options are shown in the next table.

AO_PQRS	-	AO basis integrals in Coulomb order
AO_PRQS	-	AO basis integrals in Exchange order

2-electron integrals in MO basis

In molecular basis, ORCA follows the accepted notation where by I,J,K and L we specify "internal" orbitals, meaning occupied in the reference wavefunction while by A,B,C and D we specify "external" orbitals, meaning orbitals that are empty in the reference wavefunction. With P,Q,R and S we specify all possible orbitals, meaning both "internal" and "external". The available keywords in *"orca_2json"* for the two-electron integrals in the molecular basis are the ones shown in the table below:

MO_IJKL	-	Coulomb 0-external
MO_IJKA	-	Coulomb 1-external
MO_IJAB	-	Coulomb 2-external
MO_IABC	-	Coulomb 3-external
MO_ABCD	-	Coulomb 4-external
MO_PQRS	-	Coulomb ALL integrals
MO_IKJL	-	Exchange 0-external
MO_IKJA	-	Exchange 1-external
MO_IAJB	-	Exchange 2-external
MO_IBAC	-	Exchange 3-external
MO_ACBD	-	Exchange 4-external
MO_PRQS	-	Exchange ALL integrals

Example

```
{
  "2elIntegrals":["MO_IJKL", "MO_ABCD"],
  "Thresh": 1e-8
}
```

RI 2-electron integrals

Using the Resolution of Identity (RI) one can create the integrals in a more efficient way. There are two main categories of RI integrals: the 3-index integrals, where only only half of the transformation has taken place, and the 4-index integrlas where the integrals are totally transformed in the molecular basis. The notation of the integrals follows the one we just described for the two-electron integrals in the molecular basis.

RI_IJV	-	RI 3-index 0-external
RI_IAV	-	RI 3-index 1-external
RI_ABV	-	RI 3-index 2-external
RI_IJKL	-	RI 4-index Coulomb 0-external
RI_IJKA	-	RI 4-index Coulomb 1-external
RI_IJAB	-	RI 4-index Coulomb 2-external
RI_IABC	-	RI 4-index Coulomb 3-external
RI_ABCD	-	RI 4-index Coulomb 4-external
RI_IKJL	-	RI 4-index Exchange 0-external
RI_IKJA	-	RI 4-index Exchange 1-external
RI_IAJB	-	RI 4-index Exchange 2-external
RI_IBAC	-	RI 4-index Exchange 3-external
RI_ACBD	-	RI 4-index Exchange 4-external

In addition to the integrals in case of RI integrals also the used RI Metric is available through the option Vaux:

Vaux	-	true/false

Example

```
{
  "2elIntegrals":["RI_IJKL", "RI_IJV"],
  "Vaux":true
}
```

Full Integral Transformation

The full transformation integrals can be selected via the FullTrafo keyword.

FullTrafo	-	true/false

More 2-electron integrals

Also the non-redundant 2-electron integrals are available for the RI and the nonRI case. Therefore the options 2elNonRedIntegrals or 2elNonRedRIIntegrals must be specified.

2elNonRedIntegrals	-	true/false
2elNonRedRIIntegrals	-	true/false

Orbital Windows, AuxBasisType and Threshold

The orbital window can either be selected automatically by the transformation routine or given by the user via the OrbWin keyword. The internal and external space (i0,i1,a0,a1) is defined via an integer list.

for example:

OrbWin	-	[0,8,9,15]
	-	[0,12,13,30,0,12,13,30]

The default AUX basis type is AuxC but can be changed with the keyword AuxBasisType. Please keep in mind that only those basis types used during the ORCA run can be selected.

AuxBasisType	-	AuxJ
	-	AuxJK
	-	AuxC

To reduce the number of the integrals the keyword Thresh can be used to decrease the selected integrals to save disk space. This effects ONLY the printing and not the accuracy of the generated integrals. The default print threshold is 1.e-15.

Thresh	-	printout threshold (default 1.e-15)

Example

```
{
        "OrbWin": [0,7,8,85,0,0,0,0],
        "Thresh": 1e-8,
        "AuxBasisType": "AuxC",
        "Vaux": true,
        "2elIntegrals": ["RI_IJKL"]
}
```

TDDFT amplitude data (CIS/RPA)

The TDDFT amplitudes and root informations can be requested (no triplet information yet). The available options are:

CIS	-	TDDFT amplitudes
CISNRoots	-	informations of all roots
CISROOT	-	List of roots

Example

```
{
        "CIS": true,
        "CISNRoots": false,
        "CISRoot": [1,4,7,12]
}
```

Example

```
{
        "CIS": true,
        "CISNRoots": true
}
```

JSON Format

The JSON Format that is created can also be defined in the configuration file through the *JSONFormats* variable.

JSONFormats	-	JSON output format

The available JSON formats are:

json	-	ASCII JSON format
bson	-	binary JSON format
ubjson	-	universal binary json format
msgpck	-	MessagePack format

Example

```
{
  "JSONFormats":["json","bson","ubjson","msgpck"]
}
```

MO Coefficients

MOCoefficients	-	molecule orbital information (true\false)

Example

```
{
  "MOCoefficients": true
}
```

9.3.4 Import JSON data into ORCA

Some information like geometry, basis set and Molecular orbitals stored in the json format written by orca_2json can be used to create a new gbw-file to be specified as an orbital file in the orca input. The program is called as a standalone via command line.

```
orca_2json BaseName.json -gbw
```

Basic Information

In order to create a functional gbw-file the following information must be provided:

per Atom	-	Basis
	-	Coords
	-	ElementNumber
	-	NuclearCharge
per molecule	-	Charge
	-	CoordinateUnits
	-	HFTyp
	-	Multiplicity
	-	MolecularOrbitals

NOTE

Please keep in mind that *MolecularOrbitals* is a composite of 2 different components, namely *"EnergyUnit"* and *"MOs"*. Then *"MOs"* contains *"MOCoefficients"*, *"Occupancy"*, *"OrbitalEnergy"*, "OrbitalSymLabel"* and "OrbitalSymmetry"*.

Example

The following file (let's call it *filename.json*) is a json file for H_2 molecule with STO-3G basis set.

```
{
  "Molecule": {
    "Atoms": [
      {"Basis": [
          {"Coefficients": [ 0.1543289707029839,
              0.5353281424384732,0.44463454202535485],
            "Exponents": [3.42525091,0.62391373,0.1688554],
            "Shell": "s"}],
        "Coords": [0.0,0.0,0.0],
        "ElementNumber": 1,
        "NuclearCharge": 1.0},
      {"Basis": [
          {"Coefficients": [0.1543289707029839,
              0.5353281424384732,0.44463454202535485],
            "Exponents": [3.42525091,0.62391373,0.1688554],
            "Shell": "s"}],
        "Coords": [0.0,0.0,0.8],
        "ElementNumber": 1,
        "NuclearCharge": 1.0}],
    "Charge": 0,
    "CoordinateUnits": "Angs",
    "HFTyp": "RHF",
    "MolecularOrbitals": {
      "EnergyUnit": "Eh",
      "MOs": [{"MOCoefficients": [
            -0.5554171364661243,-0.5554171364661241],
          "Occupancy": 2.0,
```

(continues on next page)

(continued from previous page)

```
          "OrbitalEnergy": -0.5544958795933514,
          "OrbitalSymLabel": "A",
          "OrbitalSymmetry": 0},
        {"MOCoefficients": [
            -1.1482994800696493,1.1482994800696493],
          "Occupancy": 0.0,
          "OrbitalEnergy": 0.6126180830925017,
          "OrbitalSymLabel": "A",
          "OrbitalSymmetry": 0}]},
    "Multiplicity": 1}
}
```

running then the command

```
orca_2json filename.json -gbw
```

should create a gbw file that ORCA can read.

Definition of the real solid harmonic Gaussian orbitals

When integrals over real solid harmonic Gaussian orbitals are issued into a JSON file, the precise definition of these orbitals becomes important. ORCA uses its own peculiar conventions for the arrangement and the phases of the individual components of the orbital shells.

The definitions of the angular parts of all shell components up to angular momentum l ($\ell = 8$) are documented below. These correspond to the real solid harmonics $S_{\ell,m}(x, y, z)$ that are normalized the following way:

$$\int_0^\pi \sin\vartheta \, d\vartheta \int_{-\pi}^{\pi} d\varphi \, r^{-2\ell} S_{\ell,m}^2(x, y, z) = 1$$

$R_\ell(r)$ is the common radial part of a shell with angular momentum ℓ that consists of a specific basis set dependent linear combination of Gaussian primitives. It is normalized independently:

$$\int_0^\infty r^{2\ell+2} R_\ell^2(r) \, dr = 1$$

The factors r^2 and $\sin\vartheta$ in these integrals arise from the volume element in spherical polar coordinates:

$$\begin{aligned} x &= r \sin\vartheta \cos\varphi \\ y &= r \sin\vartheta \sin\varphi \\ z &= r \cos\vartheta \\ dx \, dy \, dz &= r^2 \sin\vartheta \, dr \, d\vartheta \, d\varphi \end{aligned}$$

Angular momentum s ($l = 0$)

$$s = \frac{1}{2\sqrt{\pi}} R_s(r)$$

Angular momentum p ($l = 1$)

$$
\begin{aligned}
N_p &= \frac{1}{2}\sqrt{\frac{3}{\pi}} \\
p^{(0)} = p_0 \quad &= N_p\, z\, R_p(r) \\
p^{(1)} = p_{+1} &= N_p\, x\, R_p(r) \\
p^{(2)} = p_{-1} &= N_p\, y\, R_p(r)
\end{aligned}
$$

Angular momentum d ($l = 2$)

$$
\begin{aligned}
N_d &= \frac{1}{2}\sqrt{\frac{15}{\pi}} \\
d^{(0)} = d_0 \quad &= \frac{\sqrt{3}}{6}\, N_d \left(3z^2 - r^2\right) R_d(r) \\
d^{(1)} = d_{+1} &= N_d\, xz\, R_d(r) \\
d^{(2)} = d_{-1} &= N_d\, yz\, R_d(r) \\
d^{(3)} = d_{+2} &= N_d\, \frac{x^2 - y^2}{2}\, R_d(r) \\
d^{(4)} = d_{-2} &= N_d\, xy\, R_d(r)
\end{aligned}
$$

Angular momentum f ($l = 3$)

$$
\begin{aligned}
N_f &= \frac{1}{2}\sqrt{\frac{105}{\pi}} \\
f^{(0)} = f_0 \quad &= \frac{\sqrt{15}}{30}\, N_f\, z \left(5z^2 - 3r^2\right) R_f(r) \\
f^{(1)} = f_{+1} &= \frac{\sqrt{10}}{20}\, N_f\, x \left(5z^2 - r^2\right) R_f(r) \\
f^{(2)} = f_{-1} &= \frac{\sqrt{10}}{20}\, N_f\, y \left(5z^2 - r^2\right) R_f(r) \\
f^{(3)} = f_{+2} &= \frac{1}{2}\, N_f \left(x^2 - y^2\right) z\, R_f(r) \\
f^{(4)} = f_{-2} &= N_f\, xyz\, R_f(r) \\
f^{(5)} = f_{+3} &= -\frac{\sqrt{6}}{12}\, N_f\, x \left(x^2 - 3y^2\right) R_f(r) \\
f^{(6)} = f_{-3} &= -\frac{\sqrt{6}}{12}\, N_f\, y \left(3x^2 - y^2\right) R_f(r)
\end{aligned}
$$

Angular momentum g ($l = 4$)

$$
\begin{aligned}
N_g &= \frac{3}{2}\sqrt{\frac{35}{\pi}} \\
g^{(0)} = g_0 \quad &= \frac{\sqrt{35}}{280}\, N_g \left(35z^4 - 30z^2r^2 + 3r^4\right) R_g(r) \\
g^{(1)} = g_{+1} &= \frac{\sqrt{14}}{28}\, N_g\, xz \left(7z^2 - 3r^2\right) R_g(r) \\
g^{(2)} = g_{-1} &= \frac{\sqrt{14}}{28}\, N_g\, yz \left(7z^2 - 3r^2\right) R_g(r) \\
g^{(3)} = g_{+2} &= \frac{\sqrt{7}}{28}\, N_g \left(x^2 - y^2\right) \left(7z^2 - r^2\right) R_g(r) \\
g^{(4)} = g_{-2} &= \frac{\sqrt{7}}{14}\, N_g\, xy \left(7z^2 - r^2\right) R_g(r) \\
g^{(5)} = g_{+3} &= -\frac{\sqrt{2}}{4}\, N_g\, x \left(x^2 - 3y^2\right) z\, R_g(r) \\
g^{(6)} = g_{-3} &= -\frac{\sqrt{2}}{4}\, N_g\, y \left(3x^2 - y^2\right) z\, R_g(r) \\
g^{(7)} = g_{+4} &= -\frac{1}{8}\, N_g \left(x^4 - 6x^2y^2 + y^4\right) R_g(r) \\
g^{(8)} = g_{-4} &= -\frac{1}{2}\, N_g\, xy \left(x^2 - y^2\right) R_g(r)
\end{aligned}
$$

Angular momentum h ($l = 5$)

$$
\begin{aligned}
N_h &= \frac{1}{2}\sqrt{\frac{11}{\pi}} \\
h^{(0)} = h_0 \quad &= \frac{1}{8}\, N_h\, z \left(63z^4 - 70z^2r^2 + 15r^4\right) R_h(r) \\
h^{(1)} = h_{+1} &= \frac{\sqrt{15}}{8}\, N_h\, x \left(21z^4 - 14z^2r^2 + r^4\right) R_h(r) \\
h^{(2)} = h_{-1} &= \frac{\sqrt{15}}{8}\, N_h\, y \left(21z^4 - 14z^2r^2 + r^4\right) R_h(r) \\
h^{(3)} = h_{+2} &= \frac{\sqrt{105}}{4}\, N_h \left(x^2 - y^2\right) z \left(3z^2 - r^2\right) R_h(r) \\
h^{(4)} = h_{-2} &= \frac{\sqrt{105}}{2}\, N_h\, xyz \left(3z^2 - r^2\right) R_h(r) \\
h^{(5)} = h_{+3} &= -\frac{\sqrt{70}}{16}\, N_h\, x \left(x^2 - 3y^2\right) \left(9z^2 - r^2\right) R_h(r) \\
h^{(6)} = h_{-3} &= -\frac{\sqrt{70}}{16}\, N_h\, y \left(3x^2 - y^2\right) \left(9z^2 - r^2\right) R_h(r) \\
h^{(7)} = h_{+4} &= -\frac{3\sqrt{35}}{8}\, N_h \left(x^4 - 6x^2y^2 + y^4\right) z\, R_h(r) \\
h^{(8)} = h_{-4} &= -\frac{3\sqrt{35}}{2}\, N_h\, xy \left(x^2 - y^2\right) z\, R_h(r) \\
h^{(9)} = h_{+5} &= \frac{3\sqrt{14}}{16}\, N_h\, x \left(x^4 - 10x^2y^2 + 5y^4\right) R_h(r) \\
h^{(10)} = h_{-5} &= \frac{3\sqrt{14}}{16}\, N_h\, y \left(5x^4 - 10x^2y^2 + y^4\right) R_h(r)
\end{aligned}
$$

Angular momentum i ($l = 6$)

$$
\begin{aligned}
N_i &= \frac{1}{2}\sqrt{\frac{13}{\pi}} \\
i^{(0)} = i_0 \quad &= \frac{1}{16}\, N_i \left(231z^6 - 315z^4r^2 + 105z^2r^4 - 5r^6\right) R_i(r) \\
i^{(1)} = i_{+1} &= \frac{\sqrt{21}}{8}\, N_i\, xz \left(33z^4 - 30z^2r^2 + 5r^4\right) R_i(r) \\
i^{(2)} = i_{-1} &= \frac{\sqrt{21}}{8}\, N_i\, yz \left(33z^4 - 30z^2r^2 + 5r^4\right) R_i(r) \\
i^{(3)} = i_{+2} &= \frac{\sqrt{210}}{32}\, N_i \left(x^2 - y^2\right) \left(33z^4 - 18z^2r^2 + r^4\right) R_i(r) \\
i^{(4)} = i_{-2} &= \frac{\sqrt{210}}{16}\, N_i\, xy \left(33z^4 - 18z^2r^2 + r^4\right) R_i(r) \\
i^{(5)} = i_{+3} &= -\frac{\sqrt{210}}{16}\, N_i\, x \left(x^2 - 3y^2\right) z \left(11z^2 - 3r^2\right) R_i(r) \\
i^{(6)} = i_{-3} &= -\frac{\sqrt{210}}{16}\, N_i\, y \left(3x^2 - y^2\right) z \left(11z^2 - 3r^2\right) R_i(r) \\
i^{(7)} = i_{+4} &= -\frac{3\sqrt{7}}{16}\, N_i \left(x^4 - 6x^2y^2 + y^4\right) \left(11z^2 - r^2\right) R_i(r) \\
i^{(8)} = i_{-4} &= -\frac{3\sqrt{7}}{4}\, N_i\, xy \left(x^2 - y^2\right) \left(11z^2 - r^2\right) R_i(r) \\
i^{(9)} = i_{+5} &= \frac{3\sqrt{154}}{16}\, N_i\, x \left(x^4 - 10x^2y^2 + 5y^4\right) z\, R_i(r) \\
i^{(10)} = i_{-5} &= \frac{3\sqrt{154}}{16}\, N_i\, y \left(5x^4 - 10x^2y^2 + y^4\right) z\, R_i(r) \\
i^{(11)} = i_{+6} &= \frac{\sqrt{462}}{32}\, N_i \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) R_i(r) \\
i^{(12)} = i_{-6} &= \frac{\sqrt{462}}{16}\, N_i\, xy \left(3x^4 - 10x^2y^2 + 3y^4\right) R_i(r)
\end{aligned}
$$

Angular momentum k ($l = 7$)

Note that in ORCA 6.0 and earlier, basis functions of angular momentum quantum number 7 had the label j, in contrast to the usual convention in spectroscopy where the letter j is skipped. This inconsistency has been corrected

in the current release of ORCA.

$$
\begin{aligned}
N_k &= \frac{1}{2}\sqrt{\frac{15}{\pi}} \\
k^{(0)} = k_0 \quad &= \frac{1}{16}\, N_k\, z \left(429z^6 - 693z^4r^2 + 315z^2r^4 - 35r^6\right) R_k(r) \\
k^{(1)} = k_{+1} &= \frac{\sqrt{7}}{32}\, N_k\, x \left(429z^6 - 495z^4r^2 + 135z^2r^4 - 5r^6\right) R_k(r) \\
k^{(2)} = k_{-1} &= \frac{\sqrt{7}}{32}\, N_k\, y \left(429z^6 - 495z^4r^2 + 135z^2r^4 - 5r^6\right) R_k(r) \\
k^{(3)} = k_{+2} &= \frac{\sqrt{42}}{32}\, N_k \left(x^2 - y^2\right) z \left(143z^4 - 110z^2r^2 + 15r^4\right) R_k(r) \\
k^{(4)} = k_{-2} &= \frac{\sqrt{42}}{16}\, N_k\, xyz \left(143z^4 - 110z^2r^2 + 15r^4\right) R_k(r) \\
k^{(5)} = k_{+3} &= -\frac{\sqrt{21}}{32}\, N_k\, x \left(x^2 - 3y^2\right) \left(143z^4 - 66z^2r^2 + 3r^4\right) R_k(r) \\
k^{(6)} = k_{-3} &= -\frac{\sqrt{21}}{32}\, N_k\, y \left(3x^2 - y^2\right) \left(143z^4 - 66z^2r^2 + 3r^4\right) R_k(r) \\
k^{(7)} = k_{+4} &= -\frac{\sqrt{231}}{16}\, N_k \left(x^4 - 6x^2y^2 + y^4\right) z \left(13z^2 - 3r^2\right) R_k(r) \\
k^{(8)} = k_{-4} &= -\frac{\sqrt{231}}{4}\, N_k\, xy \left(x^2 - y^2\right) z \left(13z^2 - 3r^2\right) R_k(r) \\
k^{(9)} = k_{+5} &= \frac{\sqrt{231}}{32}\, N_k\, x \left(x^4 - 10x^2y^2 + 5y^4\right) \left(13z^2 - r^2\right) R_k(r) \\
k^{(10)} = k_{-5} &= \frac{\sqrt{231}}{32}\, N_k\, y \left(5x^4 - 10x^2y^2 + y^4\right) \left(13z^2 - r^2\right) R_k(r) \\
k^{(11)} = k_{+6} &= \frac{\sqrt{6006}}{32}\, N_k \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) z\, R_k(r) \\
k^{(12)} = k_{-6} &= \frac{\sqrt{6006}}{16}\, N_k\, xy \left(3x^2 - y^2\right) \left(x^2 - 3y^2\right) z\, R_k(r) \\
k^{(13)} = k_{+7} &= -\frac{\sqrt{429}}{32}\, N_k\, x \left(x^6 - 21x^4y^2 + 35x^2y^4 - 7y^6\right) R_k(r) \\
k^{(14)} = k_{-7} &= -\frac{\sqrt{429}}{32}\, N_k\, y \left(7x^6 - 35x^4y^2 + 21x^2y^4 - y^6\right) R_k(r)
\end{aligned}
$$

Angular momentum l ($l = 8$)

Note that in ORCA 6.0 and earlier, basis functions of angular momentum quantum number 8 were never actually supported. The likely result when attempting to use them was a crash.

Also note that the symbol l is reserved in basis set files for sp shells that have a common set of Gaussian exponents.

Thus for angular momentum 8 and higher, the numerical value needs to be used instead of the letter.

$$
\begin{aligned}
N_l &= \frac{1}{2}\sqrt{\frac{17}{\pi}} \\
l^{(0)} = l_0 &= \frac{1}{128} N_l \left(6435z^8 - 12012z^6r^2 + 6930z^4r^4 - 1260z^2r^6 + 35r^8\right) R_l(r) \\
l^{(1)} = l_{+1} &= \frac{3}{32} N_l\, xz \left(715z^6 - 1001z^4r^2 + 385z^2r^4 - 35r^6\right) R_l(r) \\
l^{(2)} = l_{-1} &= \frac{3}{32} N_l\, yz \left(715z^6 - 1001z^4r^2 + 385z^2r^4 - 35r^6\right) R_l(r) \\
l^{(3)} = l_{+2} &= \frac{3\sqrt{70}}{64} N_l \left(x^2 - y^2\right) \left(143z^6 - 143z^4r^2 + 33z^2r^4 - r^6\right) R_l(r) \\
l^{(4)} = l_{-2} &= \frac{3\sqrt{70}}{32} N_l\, xy \left(143z^6 - 143z^4r^2 + 33z^2r^4 - r^6\right) R_l(r) \\
l^{(5)} = l_{+3} &= -\frac{\sqrt{1155}}{32} N_l\, x \left(x^2 - 3y^2\right) z \left(39z^4 - 26z^2r^2 + 3r^4\right) R_l(r) \\
l^{(6)} = l_{-3} &= -\frac{\sqrt{1155}}{32} N_l\, y \left(3x^2 - y^2\right) z \left(39z^4 - 26z^2r^2 + 3r^4\right) R_l(r) \\
l^{(7)} = l_{+4} &= -\frac{3\sqrt{77}}{64} N_l \left(x^4 - 6x^2y^2 + y^4\right) \left(65z^4 - 26z^2r^2 + r^4\right) R_l(r) \\
l^{(8)} = l_{-4} &= -\frac{3\sqrt{77}}{16} N_l\, xy \left(x^2 - y^2\right) \left(65z^4 - 26z^2r^2 + r^4\right) R_l(r) \\
l^{(9)} = l_{+5} &= \frac{3\sqrt{1001}}{32} N_l\, x \left(x^4 - 10x^2y^2 + 5y^4\right) z \left(5z^2 - r^2\right) R_l(r) \\
l^{(10)} = l_{-5} &= \frac{3\sqrt{1001}}{32} N_l\, y \left(5x^4 - 10x^2y^2 + y^4\right) z \left(5z^2 - r^2\right) R_l(r) \\
l^{(11)} = l_{+6} &= \frac{\sqrt{858}}{64} N_l \left(x^2 - y^2\right) \left(x^4 - 14x^2y^2 + y^4\right) \left(15z^2 - r^2\right) R_l(r) \\
l^{(12)} = l_{-6} &= \frac{\sqrt{858}}{32} N_l\, xy \left(x^2 - 3y^2\right) \left(3x^2 - y^2\right) \left(15z^2 - r^2\right) R_l(r) \\
l^{(13)} = l_{+7} &= -\frac{3\sqrt{715}}{32} N_l\, x \left(x^6 - 21x^4y^2 + 35x^2y^4 - 7y^6\right) z\, R_l(r) \\
l^{(14)} = l_{-7} &= -\frac{3\sqrt{715}}{32} N_l\, y \left(7x^6 - 35x^4y^2 + 21x^2y^4 - y^6\right) z\, R_l(r) \\
l^{(15)} = l_{+8} &= -\frac{3\sqrt{715}}{128} N_l \left(x^8 - 28x^6y^2 + 70x^4y^4 - 28x^2y^6 + y^8\right) R_l(r) \\
l^{(16)} = l_{-8} &= -\frac{3\sqrt{715}}{16} N_l\, xy \left(x^2 - y^2\right) \left(x^4 - 6x^2y^2 + y^4\right) R_l(r)
\end{aligned}
$$

9.4 Property File

One of the files that ORCA produces, during a calculation, is the *property file*. The name of the file is `basename.property.txt`, where `basename` is the basename of the input file.

The property file has mainly two usages in ORCA. The first usage is to work as basis for the *Compound* scripting language. *Compound* reads all its information concerning properties through the property file and not through parsing of the ORCA output. The second usage of the property file is to make it easier for other programs, or potential GUIs, to create interfaces with ORCA.

The advantage of the property file compared to the normal ORCA output is that it's syntax is well defined and so it is easier to parse it. It can also be converted to JSON format as shown below.

Writing to the property file is enabled by default for most calculations. It can be disabled via `!NoPropFile` or toggled on/off in the `%output` block:

```
%output
  PropFile false  # default: true
end
```

Caution

Since the whole file is read and re-written multiple times for each single point calculation, optimizations of systems with a few thousand atoms can incur a few milliseconds of stacking I/O overhead, per optimization cycle (i.e., a few seconds per cycle after several hundred steps). Thus, it is recommended to disable the property file for such calculations and this is done by default for `GOAT`, `MD-L-Opt`, `MM`, or `GFN-FF` chosen in the simple input.

9.4.1 TXT Format

After any ORCA calculation a property file is created with the extension `.property.txt`. The file is a text file and one can read it and edit it with any available text editor. Below we will analyze the syntax of the file.

The file always starts with the following three lines:

```
*****************************************************
******************* ORCA 6.1.x ******************
*****************************************************
```

where, obviously the version of ORCA changes (it is also stored in `Calculation_Status -> version`).

Then, the file consists of a list of properties. Each property starts with the symbol `$` followed by the name of the property and ends with the symbol `$` followed by `End`.

For example:

```
$SCF_Nuc_Gradient
   &GeometryIndex 1
   &NAtoms [&Type "Integer"] 2
   &gradNorm [&Type "Double"]       7.6940943132805237e-02
   &grad [&Type "ArrayOfDoubles", &Dim (6,1)]
                                                          0

0                                     6.7009699964043727e-10
1                                    -1.3755030800675890e-10
2                                     5.4405462640094826e-02
3                                    -6.7009699282161039e-10
4                                     1.3755031156980307e-10
5                                    -5.4405462640095381e-02
   &Method [&Type "String"] "SCF"
   &Level [&Type "String"] "Relaxed density"
   &Mult [&Type "Integer"] 1
   &State [&Type "Integer"] 0
   &Irrep [&Type "Integer"] 0
$End
```

Each property consists of *components*. Each component starts with the symbol `&` and has no ending symbol.

For example:

```
   &gradNorm [&Type "Double"]       7.6940943132805237e-02
```

The first component in every property is `GeometryIndex`, which shows the geometry to which the current property belongs. For example:

```
&GeometryIndex 1
```

Note that geometry indices start from 1.

The remaining components have the following syntax:

- First the start of component symbol `&`.
- Then follows the name of the component.
- Then a bracket opens with various bracket information about the component. For details on the syntax of the bracket information please check *Bracket Information*.

After the bracket there are different options.

If the type is `String` then a string is expected starting with quotation marks. For example:

```
&Method [&Type "String"] "SCF"
```

If the type is `Double` or `Integer` then a number of the appropriate type is expected. For example:

```
&NAtoms [&Type "Integer"] 2 "Number of atoms"
```

As in the example above, a component description or comment in quotation marks may be optionally present after the value in the same line as the component name. Note that for `Array` components, the comment actually comes *before* the data.

Finally, if the type is a kind of `Array` then, after the optional comment, an array is written starting from the next line. Arrays are written in blocks of up to 8 whitespace-sparated columns. Each block starts with a line with column indices (starting from 0). The second line in a block is reserved but currently ignored. The next lines start with the row index and continue with the data for the respective row/columns. For example:

```
   &dipoleTotal [&Type "ArrayOfDoubles", &Dim (3,1)] "Total"
                                                          0

0                                     -5.1716015643064861e-01
1                                      5.1774970936823518e-02
2                                      7.8324050140722279e-01
```

Note that `Complex` scalars are printed as two consecutive values for the real and imaginary part (before the optional comment), whereas for `ArrayOfComplex` the real part of the whole array is printed first, followed by the imaginary part.

Bracket Information

Bracket information is a list of entries separated with ‘,’. The first and most important bracket entry is the `Type`, which can be one of the following:

- `Double`
- `Integer`
- `Boolean`
- `Complex`
- `String`
- `ArrayOfDoubles`
- `ArrayOfIntegers`
- `ArrayOfBooleans`
- `ArrayOfComplex`
- `Coordinates`

For example:

```
&NAtoms [&Type "Integer"] 2
```

Then, in case the `Type` is a kind of array, the bracket must contain the dimensions of the array, using the `Dim` entry. Note that all arrays are 2-dimensional! For example:

```
&ATNO [&Type "ArrayOfIntegers", &Dim (2,1)]
```

Optionally, `Units` may also be given, if the value is not in a.u. For example:

```
&dipoleMagnitude [&Type "Double", &Units "Debye"]       9.4000050960598935e-01
```

9.4.2 JSON Format

The property file can be also produced in a JSON format. Internally this happens through transformation of the `txt` format to JSON format. There are two ways to create a JSON property file.

The first way is through the normal ORCA input using the `JSONPropFile` keyword.

```
%Output
  JSONPropFile True
End
```

this will create a `basename.property.txt` and in addition a `basename.property.json` file.

The second way is through the *`orca_2json` command*. For this, one first has to run a normal ORCA input, that will create a `basename.property.txt` file, and then convert it to `basename.property.json` using the command:

```
orca_2json basename -property
```

The file is structured as follows:

```
{
  "<GlobalPropertyName>": {
    "<ComponentName>": <value>,
    <More components>
  },
  <More global properties>
  "Geometries": [
    {
      "Geometry": {
        "Coordinates": {
            "Cartesians": [
                [
                    "<Symbol>",
                    <X>,
                    <Y>,
                    <Z>
                ],
                <More atoms>
            ],
            "Type": "Cartesians",
            "Units": "a.u."
        },
        "NAtoms": <value>,
        <More components>
      },
      "<PropertyName>": {
        "<ComponentName>": <value>,
        <More components>
```

(continues on next page)

(continued from previous page)

```
      },
      <More properties for this geometry>
    },
    <More geometries>
  ]
}
```

The "global", i.e. geometry-independent properties are: `Calculation_Status`, `Calculation_Info`, and `Calculation_Timings`.

Most component data types are directly converted to the corresponding JSON types. All `Array` components are stored as two-dimensional using nested lists (even if one dimension is of length 1). `Complex` and `ArrayOfComplex` values are stored as separate real and imaginary parts:

```
"ComplexComponentName": {
  "re": <real part>,
  "im": <imaginary part>
},
"ComplexArrayComponentName": {
  "re": [
      [<r11>, <r12>, <...>],
      [<r21>, <r22>, <...>],
      <...>
    ],
  "im": [
      [<i11>, <i12>, <...>],
      [<i21>, <i22>, <...>],
      <...>
    ]
}
```

9.4.3 List of Available Properties

Table 9.1: List of available properties

Property[1]	M[2]	Component	T[3]	O[4]	L[5]	Description	Units
`AutoCI_Energies` [`Energy`]	Y	`numOfEl`	I	N	N		
		`numOfCorrEl`	I	N	N		
		`numOfAlphaCorrEl`	I	Y	N		
		`numOfBetaCorrEl`	I	Y	N		
		`refEnergy`	AD	N	N	Reference energy for each state	
		`corrEnergy`	AD	N	N	Total correlation energy for each state	
		`Method`	S	N	N		
		`totalEnergy`	AD	N	N	Total energy of each state	
		`Mult`	AI	Y	N	Multiplicity of each state	
		`Irrep`	AI	Y	N	Irreducible representation of each state	
		`RelCorr`	S	Y	N	Relativistic correction (SOC and/or SSC)	
		`NBlocks`	I	Y	N	Number of multiplicity blocks	
		`NRoots`	AI	Y	N	Number of roots in each block	
		`NTotalRoots`	I	Y	N	Total number of roots	
		`Block`	AI	Y	N	Block index of each state	
		`Root`	AI	Y	N	Root index within the block	
		`FollowIRoot`	I	Y	N	Index of the followed root	

continues on next page

Table 9.1 – continued from previous page

		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
AutoCI_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
EFG_Tensor	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
AutoCI_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
AutoCI_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
AutoCI_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	

continues on next page

Table 9.1 – continued from previous page

		Isotope	D	N	Y	Atomic mass
		I	D	N	Y	Spin of the nuclei
		PFAC	D	N	Y	Prefactor PFAC=ge*g*Nbe*bN (in MHz)
		ARaw	AD	N	Y	Raw tensor
		AEigenvalues	AD	N	Y	Eigenvalues
		orientation	AD	N	Y	Eigenvectors
		AIso	D	N	Y	
AUTOCI_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
AutoCI_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
AutoCI_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
AutoCI_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.

continues on next page

Table 9.1 – continued from previous page

		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
AutoCI_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
AutoCI_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
AutoCI_MRCI_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
AutoCI_MRCI_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CAS_DCD_Energies [Energy]	Y	finalEnergy	D	N	N	Final GS or SA energy
		numOfElectrons	I	N	N	
		numOfActiveEl	I	N	N	Number of active electrons
		numOfActiveOrbs	I	N	N	Number of active orbitals
		numOfFCElectrons	I	N	N	
		Method	S	N	N	
		totalEnergy	AD	N	N	Total energy of each state
		Mult	AI	Y	N	Multiplicity of each state
		Irrep	AI	Y	N	Irreducible representation of each state
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)
		NBlocks	I	Y	N	Number of multiplicity blocks
		NRoots	AI	Y	N	Number of roots in each block
		NTotalRoots	I	Y	N	Total number of roots
		Block	AI	Y	N	Block index of each state
		Root	AI	Y	N	Root index within the block
		FollowIRoot	I	Y	N	Index of the followed root
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state
CAS_MSPT2_Energies [Energy]	Y	finalEnergy	D	N	N	Final GS or SA energy
		numOfElectrons	I	N	N	
		numOfActiveEl	I	N	N	Number of active electrons
		numOfActiveOrbs	I	N	N	Number of active orbitals
		numOfFCElectrons	I	N	N	
		Method	S	N	N	
		totalEnergy	AD	N	N	Total energy of each state
		Mult	AI	Y	N	Multiplicity of each state

continues on next page

Table 9.1 – continued from previous page

		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
CAS_PT2_Energies [Energy]	Y	finalEnergy	D	N	N	Final GS or SA energy	
		numOfElectrons	I	N	N		
		numOfActiveEl	I	N	N	Number of active electrons	
		numOfActiveOrbs	I	N	N	Number of active orbitals	
		numOfFCElectrons	I	N	N		
		Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
CAS_SCF_Energies [Energy]	Y	finalEnergy	D	N	N	Final GS or SA energy	
		numOfElectrons	I	N	N		
		numOfActiveEl	I	N	N	Number of active electrons	
		numOfActiveOrbs	I	N	N	Number of active orbitals	
		numOfFCElectrons	I	N	N		
		Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
CASSCF_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	

continues on next page

Table 9.1 – continued from previous page

		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CASSCF_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
CASSCF_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CASSCF_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CASSCF_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	

continues on next page

Table 9.1 – continued from previous page

		ARaw	AD	N	Y	Raw tensor
		AEigenvalues	AD	N	Y	Eigenvalues
		orientation	AD	N	Y	Eigenvectors
		AIso	D	N	Y	
CASSCF_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
CASSCF_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CASSCF_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CASSCF_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	

continues on next page

Table 9.1 – continued from previous page

		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_CASSCF_2ndOrder [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_CASSCF_2ndOrder_ES [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_CASSCF_Heff [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_CASSCF_Heff_ES [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_NEVPT2_2ndOrder [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_NEVPT2_2ndOrder_ES [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N
		Mult	I	N	N
		State	I	N	N
		Irrep	I	N	N
D_Tensor_NEVPT2_Heff [D_Tensor]	Y	d_raw	AD	N	N
		d_eigenvalues	AD	N	N
		d_eigenvectors	AD	N	N
		D	D	N	N
		E	D	N	N
		Method	S	N	N
		Level	S	N	N

continues on next page

Table 9.1 – continued from previous page

		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_NEVPT2_Heff_ES [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_CUSTOM_2ndOrder [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_CUSTOM_2ndOrder_ES [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_CUSTOM_Heff [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_CUSTOM_Heff_ES [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_CASSCF_2ndOrder [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_CASSCF_Heff [G_Tensor]	Y	g_matrix	AD	N	N	

continues on next page

Table 9.1 – continued from previous page

		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_NEVPT2_2ndOrder [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_NEVPT2_Heff [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_CUSTOM_2ndOrder [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
G_Tensor_CUSTOM_Heff [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution

continues on next page

Table 9.1 – continued from previous page

		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CASSCF_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_PT_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin

continues on next page

Table 9.1 – continued from previous page

		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_QDPT_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_DCD_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	

continues on next page

Table 9.1 – continued from previous page

	RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
	DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
	DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
	DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
	Density_name	S	N	N	
	Representation	S	N	N	Possible values: Unknown, Length, Velocity
	PointGroup	S	N	N	
	DoHigherMoments	B	N	N	
	NTrans	I	N	N	
	ExcitationEnergies	AD	N	N	
	States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
	Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
	Temperature	D	N	N	
CASSCF_PT_Energies_Absorption_Spectrum [Absorption_Spectrum]	Method	I	N	N	
	RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
	DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
	DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
	DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
	Density_name	S	N	N	
	Representation	S	N	N	Possible values: Unknown, Length, Velocity
	PointGroup	S	N	N	
	DoHigherMoments	B	N	N	
	NTrans	I	N	N	

continues on next page

Table 9.1 – continued from previous page

		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_Custom_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
CASSCF_PT_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
CASSCF_QDPT_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
CASSCF_DCD_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		

continues on next page

Table 9.1 – continued from previous page

		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_PT_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CASSCF_Custom_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
CIPSI_Energies	N	finalEnergy	D	N	N		
		numOfRoots	I	N	N		
		multiplicity	I	N	N		
		energies	AD	N	N		
CIS_Energies [Energy]	Y	e0	D	N	N	Ground state energy	
		multP1	B	N	N	Do the higher multiplicity too?	
		mode	S	N	N	CIS mode: CIS, RPA, TDA, TD-DFT, sTDA, sTD-DFT	
		dCorr	I	N	N	(D) Correction algorithm	
		Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
CIS_Nuc_Gradient [Nuclear_Gradient]	Y	NAtoms	I	N	N		
		gradNorm	D	N	N		
		grad	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CIS_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		Irrep	I	N	N		
CIS_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
CIS_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CIS_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CIS_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*g*Nbe*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
CIS_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
CIS_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CIS_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CIS_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CIS_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	

continues on next page

Table 9.1 – continued from previous page

		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
CIS_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	

continues on next page

Table 9.1 – continued from previous page

		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
DBOC_Energy	N	DBOC_Energy	D	N	N		
DFT_Energy	N	nAlphaEl	I	N	N		
		nBetaEl	I	N	N		
		nTotalEl	I	N	N		
		finalEn	D	N	N	No Van der Waals correction	
		eExchange	D	N	N		
		eCorr	D	N	N		
		eXC	D	N	N		
		eCNL	D	N	N		
		eEmbed	D	Y	N		
BROKEN_SYMMETRY	N	enHighSpin	D	N	N	The High Spin Energy	
		enBrokenSym	D	N	N	The Broken Symmetry Energy	
		SHighSpin	D	N	N	The High Spin Spin	
		S2HighSpin	D	N	N	The Expectation value of S**2 for the High Spin case	
		S2BrokenSym	D	N	N	The Expectation value of S**2 for the Broken symmetry case	
Dipole_Moment	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
EFG_Tensor	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
Quadrupole_Moment	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		

continues on next page

Table 9.1 – continued from previous page

		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
Polarizability	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
Hyperpolarizability	Y	rawCartesian	AD	N	N		a.u
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
Excited_States_Dynamics	N	fluorRateConstant	D	N	N		
Energy_Extrapolation	N	doEp1	B	N	N	Extrapolation using only one method	
		doEp2	B	N	N	Extrapolation using two different methods (same small basis Set)	
		doEp3	B	N	N	Extrapolation using two different methods (use three basis Sets for the cheap method)	
		doGradients	B	N	N	Extrapolate energy gradients	
		scfEnergies	AD	N	N		
		scfCBS	D	N	N		
		scfGradients	AD	Y	N	The SCF Gradients	
		corrEnergies	AD	N	N		
		corrCBS	D	N	N		
		ccsdtEnergyX	D	Y	N		
		totalCBS	D	N	N		
		cardinalNumbers	AI	N	N		
		alphas	AD	N	N		
		betas	AD	N	N		
		numOfEnergies	I	N	N	How many energies we are going to use (e.g. two-point scheme).	
		basisName	S	N	Y		
gCP_Energy	N	gCP_Energy	D	N	N		
Hessian	N	HESSIAN	AD	N	N	The Hessian	
		MODES	AD	N	N	The modes of vibrations	
ICE_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ICE_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
ICE_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ICE_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ICE_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
ICE_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	

continues on next page

Table 9.1 – continued from previous page

		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
ICE_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
ICE_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
ICE_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
ICE_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
ICE_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	

continues on next page

Table 9.1 – continued from previous page

		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
Calculation_Info	N	Mult	I	N	N	
		Charge	I	N	N	
		NumOfAtoms	I	N	N	
		NumOfElectrons	I	N	N	
		NumOfBasisFuncts	I	N	N	
		NumOfAuxCBasisFuncts	I	N	N	
		NumOfAuxJBasisFuncts	I	N	N	
		NumOfAuxJKBasisFunct	I	N	N	
		NumOfCABSBasisFuncts	I	N	N	
Single_Point_Data	N	FinalEnergy	D	N	N	Final single point energy
		Converged	B	N	N	
LFT_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
LFT_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
A_Tensor	Y	numOfNucs	I	N	N	Number of active nuclei
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elem	I	N	Y	Atomic number of the nuclei
		Isotope	D	N	Y	Atomic mass
		I	D	N	Y	Spin of the nuclei
		PFAC	D	N	Y	Prefactor PFAC=ge*g*Nbe*bN (in MHz)
		ARaw	AD	N	Y	Raw tensor
		AEigenvalues	AD	N	Y	Eigenvalues
		orientation	AD	N	Y	Eigenvectors
		AIso	D	N	Y	
Chemical_Shift	Y	numOfNucs	I	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	

continues on next page

Table 9.1 – continued from previous page

D_Tensor	Y	d_raw	AD	N	N		
		d_eigenvalues	AD	N	N		
		d_eigenvectors	AD	N	N		
		D	D	N	N		
		E	D	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
G_Tensor	Y	g_matrix	AD	N	N		
		g_elec	D	N	N	The free electron g-value contribution	
		g_RMC	D	N	N	The reduced mass correction	
		g_DSO	AD	N	N		
		g_PSO	AD	N	N		
		g_Tot	AD	N	N		
		g_iso	D	N	N		
		Delta_g	AD	N	N		
		Delta_g_iso	D	N	N		
		orientation	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
Spin_Spin_Coupling	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate	
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO	
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO	
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC	
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD	
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC	
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.	
		pairsDistances	AD	N	N	The distances of each pair	
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MCRPA_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MCRPA_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
MCRPA_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MCRPA_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MCRPA_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
MCRPA_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		SDSO	AD	N	Y	Diamagnetic contribution	

continues on next page

Table 9.1 – continued from previous page

		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
MCRPA_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MCRPA_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MCRPA_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MCRPA_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
MCRPA_ECD_Spectrum[ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		

continues on next page

Table 9.1 – continued from previous page

		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
MDCI_Energies [Energy]	Y	numOfEl	I	N	N		
		numOfCorrEl	I	N	N		
		numOfAlphaCorrEl	I	N	N		
		numOfBetaCorrEl	I	N	N		
		refEnergy	AD	N	N	Reference Energy	
		corrEnergy	AD	N	N	Total Correlation Energy	
		aaCorrEn	AD	N	N	Alpha-Alpha Pairs Correlation Energy (No (T))	
		bbCorrEn	AD	N	N	Beta-Beta Pairs Correlation Energy (No (T))	
		abCorrEn	AD	N	N	Alpha-Beta Pairs Correlation Energy (No (T))	
		CorrDS	AD	Y	N	Singlet pairs energy of double amplitudes (No (T))	
		CorrDT	AD	Y	N	Triplet pairs energy of double amplitudes (No (T))	
		CorrSS	AD	Y	N	Singlet pairs energy of quadratic single amplitudes (No (T))	
		CorrST	AD	Y	N	Triplet pairs energy of quadratic single amplitudes (No (T))	
		triplesEnergy	AD	N	N	Triples Correction Energy	
		Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
MDCI_Nuc_Gradient [Nuclear_Gradient]	Y	NAtoms	I	N	N		
		gradNorm	D	N	N		
		grad	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MDCI_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		

continues on next page

Table 9.1 – continued from previous page

		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MDCI_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
MDCI_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MDCI_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MDCI_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
MDCI_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		

continues on next page

Table 9.1 – continued from previous page

		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
MDCI_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MDCI_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MDCI_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MDCI_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	

continues on next page

Table 9.1 – continued from previous page

		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
MDCI_ECD_Spectrum[ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	

continues on next page

Table 9.1 – continued from previous page

		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
MP2_Energies [Energy]	Y	refEnergy	AD	N	N	Reference energy for each state	
		corrEnergy	AD	N	N	MP2 correlation energy for each state	
		Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
MP2_Nuc_Gradient [Nuclear_Gradient]	Y	NAtoms	I	N	N		
		gradNorm	D	N	N		
		grad	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MP2_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MP2_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
MP2_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		

continues on next page

Table 9.1 – continued from previous page

		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MP2_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MP2_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
MP2_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		SDSO	AD	N	Y	Diamagnetic contribution	
		SPSO	AD	N	Y	Paramagnetic contribution	
		STot	AD	N	Y	Total tensor	
		orientation	AD	N	Y	Eigenvectors	
		sTotEigen	AD	N	Y	Eigenvalues	
		siso	D	N	Y		
		saniso	D	N	Y		
MP2_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N		
		d_eigenvalues	AD	N	N		
		d_eigenvectors	AD	N	N		
		D	D	N	N		
		E	D	N	N		

continues on next page

Table 9.1 – continued from previous page

		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MP2_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MP2_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MP2_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
MP2_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
MRCI_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		

continues on next page

Table 9.1 – continued from previous page

		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MRCI_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
MRCI_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MRCI_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
MRCI_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
MRCI_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		

continues on next page

Table 9.1 – continued from previous page

		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elems	I	N	Y	Atomic number of the nuclei
		SDSO	AD	N	Y	Diamagnetic contribution
		SPSO	AD	N	Y	Paramagnetic contribution
		STot	AD	N	Y	Total tensor
		orientation	AD	N	Y	Eigenvectors
		sTotEigen	AD	N	Y	Eigenvalues
		siso	D	N	Y	
		saniso	D	N	Y	
MRCI_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MRCI_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MRCI_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_MRCI_2ndOrder [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	

continues on next page

Table 9.1 – continued from previous page

		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_MRCI_2ndOrder_ES [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_MRCI_Heff [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
D_Tensor_MRCI_Heff_ES [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MRCI_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	

continues on next page

Table 9.1 – continued from previous page

		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
MRCI_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
Nuclear_Gradient	Y	NAtoms	I	N	N	
		gradNorm	D	N	N	
		grad	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
CHELPG_Population_Analysis	Y	NAtoms	I	N	N	
		ATNO	AI	N	N	
		AtomicCharges	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
Hirshfeld_Population_Analysis	Y	NAtoms	I	N	N	
		ATNO	AI	N	N	
		DENSA	D	N	N	Total integrated alpha density

continues on next page

Table 9.1 – continued from previous page

		DENSB	D	N	N	Total integrated beta density
		AtomicCharges	AD	N	N	
		Spin	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
Loewdin_Population_Analysis	Y	NAtoms	I	N	N	
		ATNO	AI	N	N	
		AtomicCharges	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
Mayer_Population_Analysis	Y	NAtoms	I	N	N	Total number of atoms
		NBondOrdersPrint	I	N	N	The number of bond orders larger than threshold
		bondThresh	D	N	N	The threshold for printing
		components	AI	Y	N	The indices and atomic numbers of the bonding atoms
		BondOrders	AD	Y	N	The bond orders
		ATNO	AI	N	N	Atomic number of the elements
		NA	AD	N	N	Mulliken gross atomic population
		ZA	AD	N	N	Total nuclear charge
		QA	AD	N	N	Mulliken gross atomic charge
		VA	AD	N	N	Mayer's total valence
		BVA	AD	N	N	Mayer's bonded valence
		FA	AD	N	N	Mayer's free valence
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MBIS_Population_Analysis	Y	NAtoms	I	N	N	
		ATNO	AI	N	N	
		Thresh	D	N	N	
		Niter	I	N	N	
		LargePrint	B	N	N	
		DENSA	D	N	N	Total integrated alpha density
		DENSB	D	N	N	Total integrated beta density
		AtomicCharges	AD	N	N	
		Spin	AD	N	N	
		NPOPVAL	AD	N	N	
		SIGMAVAL	AD	N	N	
		AtomicDipole	AD	Y	N	
		AtomicQuadrupole	AD	Y	N	
		AtomicOctupole	AD	Y	N	
		ThirdRadialMoment	AD	Y	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
Mulliken_Population_Analysis	Y	NAtoms	I	N	N	
		ATNO	AI	N	N	
		AtomicCharges	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	

continues on next page

Table 9.1 – continued from previous page

RASCI_Nuc_Gradient [Nuclear_Gradient]	Y	NAtoms	I	N	N		
		gradNorm	D	N	N		
		grad	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RASCI_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RASCI_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
RASCI_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RASCI_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RASCI_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		State	I	N	N	
		Irrep	I	N	N	
		NUC	I	N	Y	Index of the nuclei
		Elem	I	N	Y	Atomic number of the nuclei
		Isotope	D	N	Y	Atomic mass
		I	D	N	Y	Spin of the nuclei
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)
		ARaw	AD	N	Y	Raw tensor
		AEigenvalues	AD	N	Y	Eigenvalues
		orientation	AD	N	Y	Eigenvectors
		AIso	D	N	Y	
RASCI_Chemical_Shift [Chemical_Shift]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RASCI_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N	
		d_eigenvalues	AD	N	N	
		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RASCI_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RASCI_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair

continues on next page

Table 9.1 – continued from previous page

		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RASCI_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
RASCI_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
ROCIS_Energies	N	refEnergy	D	N	N		
		corrEnergy	D	N	N		
		totalEnergy	D	N	N		
		numOfRoots	I	N	N		
		Energies	AD	N	N	State energies in Hartree	
		Multiplicities	AI	N	N		
ROCIS_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ROCIS_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
ROCIS_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ROCIS_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	

continues on next page

Table 9.1 – continued from previous page

		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ROCIS_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
ROCIS_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		SDSO	AD	N	Y	Diamagnetic contribution	
		SPSO	AD	N	Y	Paramagnetic contribution	
		STot	AD	N	Y	Total tensor	
		orientation	AD	N	Y	Eigenvectors	
		sTotEigen	AD	N	Y	Eigenvalues	
		siso	D	N	Y		
		saniso	D	N	Y		
ROCIS_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N		
		d_eigenvalues	AD	N	N		
		d_eigenvectors	AD	N	N		
		D	D	N	N		
		E	D	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
ROCIS_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N		
		g_elec	D	N	N	The free electron g-value contribution	
		g_RMC	D	N	N	The reduced mass correction	
		g_DSO	AD	N	N		
		g_PSO	AD	N	N		
		g_Tot	AD	N	N		
		g_iso	D	N	N		
		Delta_g	AD	N	N		

continues on next page

Table 9.1 – continued from previous page

		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
ROCIS_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
ROCIS_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep

continues on next page

Table 9.1 – continued from previous page

		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
ROCIS_ECD_Spectrum[ECD_Spectrum]	Y	Method	I	N	N		
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC	
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin	
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad	
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed	
		Density_name	S	N	N		
		Representation	S	N	N	Possible values: Unknown, Length, Velocity	
		PointGroup	S	N	N		
		DoHigherMoments	B	N	N		
		NTrans	I	N	N		
		ExcitationEnergies	AD	N	N		
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
RPA_Dipole_Moment[Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RPA_EFG_Tensor[EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	

continues on next page

Table 9.1 – continued from previous page

		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
RPA_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		
		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RPA_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
RPA_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*g*Nbe*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
RPA_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		SDSO	AD	N	Y	Diamagnetic contribution	
		SPSO	AD	N	Y	Paramagnetic contribution	
		STot	AD	N	Y	Total tensor	
		orientation	AD	N	Y	Eigenvectors	
		sTotEigen	AD	N	Y	Eigenvalues	
		siso	D	N	Y		
		saniso	D	N	Y		
RPA_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N		
		d_eigenvalues	AD	N	N		

continues on next page

Table 9.1 – continued from previous page

		d_eigenvectors	AD	N	N	
		D	D	N	N	
		E	D	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RPA_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RPA_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
RPA_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad

continues on next page

Table 9.1 – continued from previous page

		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
RPA_ECD_Spectrum [ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
Energy	Y	Method	S	N	N	
		totalEnergy	AD	N	N	Total energy of each state
		Mult	AI	Y	N	Multiplicity of each state
		Irrep	AI	Y	N	Irreducible representation of each state
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)
		NBlocks	I	Y	N	Number of multiplicity blocks

continues on next page

Table 9.1 – continued from previous page

		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
SCF_Energy [Energy]	Y	Method	S	N	N		
		totalEnergy	AD	N	N	Total energy of each state	
		Mult	AI	Y	N	Multiplicity of each state	
		Irrep	AI	Y	N	Irreducible representation of each state	
		RelCorr	S	Y	N	Relativistic correction (SOC and/or SSC)	
		NBlocks	I	Y	N	Number of multiplicity blocks	
		NRoots	AI	Y	N	Number of roots in each block	
		NTotalRoots	I	Y	N	Total number of roots	
		Block	AI	Y	N	Block index of each state	
		Root	AI	Y	N	Root index within the block	
		FollowIRoot	I	Y	N	Index of the followed root	
		AvgMult	AD	Y	N	Average multiplicity of each SO-/SS-coupled state	
SCF_Nuc_Gradient [Nuclear_Gradient]	Y	NAtoms	I	N	N		
		gradNorm	D	N	N		
		grad	AD	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
SCF_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
SCF_EFG_Tensor [EFG_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		QFAC	D	N	Y	Prefactor	
		V	AD	N	Y	Raw tensor	
		VEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		VIso	D	N	Y		
SCF_Polarizability [Polarizability]	Y	isotropicPolar	D	N	N		
		rawCartesian	AD	N	N		a.u
		diagonalizedTensor	AD	N	N		
		orientation	AD	N	N		

continues on next page

Table 9.1 – continued from previous page

		doAtomicPolar	B	N	N		
		atomicPolarIso	AD	Y	N	Atomic istotropic polarizabilities	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
SCF_Quadrupole_Moment [Quadrupole_Moment]	Y	isotropicQuadMoment	D	N	N		a.u.
		quadElecContrib	AD	N	N	Electronic contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadNucContrib	AD	N	N	Nuclear contribution. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadTotal	AD	N	N	Total. Order: XX, YY, ZZ, XY, XZ, YZ	
		quadDiagonalized	AD	N	N	The diagonalized tensor	a.u.
		doAtomicQuad	B	N	N		
		atomicQuad	AD	Y	N	Atomic quadrupoles (NAtoms * XX, YY, ZZ, XY, XZ, YZ)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
SCF_A_Tensor [A_Tensor]	Y	numOfNucs	I	N	N	Number of active nuclei	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elem	I	N	Y	Atomic number of the nuclei	
		Isotope	D	N	Y	Atomic mass	
		I	D	N	Y	Spin of the nuclei	
		PFAC	D	N	Y	Prefactor PFAC=ge*gN*be*bN (in MHz)	
		ARaw	AD	N	Y	Raw tensor	
		AEigenvalues	AD	N	Y	Eigenvalues	
		orientation	AD	N	Y	Eigenvectors	
		AIso	D	N	Y		
SCF_Chemical_Shift [Chemical_Shift]	Y	numOfNucs	I	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
		NUC	I	N	Y	Index of the nuclei	
		Elems	I	N	Y	Atomic number of the nuclei	
		SDSO	AD	N	Y	Diamagnetic contribution	
		SPSO	AD	N	Y	Paramagnetic contribution	
		STot	AD	N	Y	Total tensor	
		orientation	AD	N	Y	Eigenvectors	
		sTotEigen	AD	N	Y	Eigenvalues	
		siso	D	N	Y		
		saniso	D	N	Y		
SCF_D_Tensor [D_Tensor]	Y	d_raw	AD	N	N		
		d_eigenvalues	AD	N	N		
		d_eigenvectors	AD	N	N		
		D	D	N	N		
		E	D	N	N		
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		

continues on next page

Table 9.1 – continued from previous page

		State	I	N	N	
		Irrep	I	N	N	
SCF_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N	
		g_elec	D	N	N	The free electron g-value contribution
		g_RMC	D	N	N	The reduced mass correction
		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
SCF_Spin_Spin_Coupling [Spin_Spin_Coupling]	Y	numOfNucPairs	I	N	N	Number of nuclei pairs to calculate
		numOfNucPairsDSO	I	N	N	number of nuclear pairs to calculate DSO
		numOfNucPairsPSO	I	N	N	number of nuclear pairs to calculate PSO
		numOfNucPairsFC	I	N	N	number of nuclear pairs to calculate FC
		numOfNucPairsSD	I	N	N	number of nuclear pairs to calculate SD
		numOfNucPairsSD_FC	I	N	N	number of nuclear pairs to calculate SD/FC
		pairsInfo	AI	N	N	Pairs Info: Col1->Index of A. Col2->Atom. Num. of A. Col3->Index of B. Col4->Atom. Num. of B.
		pairsDistances	AD	N	N	The distances of each pair
		pairsTotalSSCIso	AD	N	N	The Spin-Spin coupling constant for each pair
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
SCF_Absorption_Spectrum [Absorption_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	

continues on next page

Table 9.1 – continued from previous page

		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
SCF_ECD_Spectrum[ECD_Spectrum]	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
SOC_Energy_Correction	N	total_SOC_energy	D	N	N	
		total_non_SOC_energy	D	N	N	
		nuclear_energy	D	N	N	
		SOC_correction	D	N	N	
Absorption_Spectrum	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC

continues on next page

Table 9.1 – continued from previous page

		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	
		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state
		Temperature	D	N	N	
ECD_Spectrum	Y	Method	I	N	N	
		RelCorrection	I	N	N	Type of relativistic treatment in QDPT. 0->Unknow, 1->None, 2->SOC, 3->SSC, 4->SOC/SSC
		DensType	I	N	N	Type of density (electron/spin …). 0->Unknow, 1->Electronic, 2->Spin, 3->Trans/Electronic, 4->Tran/Spin
		DeriType	I	N	N	Type of derivative (w.r.t. to what perturbation). 0->Unknown, 1->No Derivative, 2->Electric/Dipole, 3->Electric/Quad, 4->Magnetic/Dipole, 5->Magnetic/Quad
		DensLevel	I	N	N	// Source of density: relaxed, unrelaxed, … 0->Unknown, 1->Linearized, 2->Unrelaxed, 3->Relaxed
		Density_name	S	N	N	
		Representation	S	N	N	Possible values: Unknown, Length, Velocity
		PointGroup	S	N	N	
		DoHigherMoments	B	N	N	
		NTrans	I	N	N	
		ExcitationEnergies	AD	N	N	

continues on next page

Table 9.1 – continued from previous page

		States	AI	N	N	The initial and Final states. Col1: Initial State Col2: Initial Irrep Col3: Final State Col4: Final Irrep	
		Multiplicities	AD	N	N	Col1: Multiplicity of the initial state Col2: Multiplicity of the final state	
		Temperature	D	N	N		
THERMOCHEMISTRY_Energies	Y	temperature	D	N	N		
		pressure	D	N	N		
		totalMass	D	N	N		
		spinDegeneracy	I	N	N		
		elEnergy	D	N	N		
		transEnergy	D	N	N		
		rotEnergy	D	N	N		
		vibEnergy	D	N	N		
		numOfFreqs	I	N	N		
		freqScalingFactor	D	N	N		
		FREQ	AD	N	N		cm^-1
		zpe	D	N	N		
		innerEnergyU	D	N	N		
		enthalpyH	D	N	N		
		qEl	D	N	N		
		qRot	D	N	N		
		qVib	D	N	N		
		qTrans	D	N	N		
		entropyS	D	N	N		
		freeEnergyG	D	N	N		
		isLinear	B	N	N		
VdW_Correction	N	vdW	D	N	N		
		vdW_atomic	AD	Y	N		
Calculation_Status	N	version	S	N	N		
		progName	S	N	N		
		Status	S	N	N		
MDCI_EOM_Energies	Y	groundRefEnergy	D	N	N	Ground State Reference Energy	
		groundCorrEnergy	D	N	N	Ground State Correlation Energy	
		groundTotalEnergy	D	N	N	Ground State Total MDCI Energy	
CASPT2_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CAS_CUSTOM_Dipole_Moment [Dipole_Moment]	Y	dipoleMagnitude	D	N	N		a.u.
		dipoleElecContrib	AD	N	N	Electronic contribution	
		dipoleNucContrib	AD	N	N	Nuclear contribution	
		dipoleTotal	AD	N	N	Total	
		doAtomicDipole	B	N	N		
		atomicDipole	AD	Y	N	Atomic dipoles (NAtoms * X,Y,Z)	
		Method	S	N	N		
		Level	S	N	N		
		Mult	I	N	N		
		State	I	N	N		
		Irrep	I	N	N		
CASPT2_G_Tensor [G_Tensor]	Y	g_matrix	AD	N	N		
		g_elec	D	N	N	The free electron g-value contribution	
		g_RMC	D	N	N	The reduced mass correction	

continues on next page

Table 9.1 – continued from previous page

		g_DSO	AD	N	N	
		g_PSO	AD	N	N	
		g_Tot	AD	N	N	
		g_iso	D	N	N	
		Delta_g	AD	N	N	
		Delta_g_iso	D	N	N	
		orientation	AD	N	N	
		Method	S	N	N	
		Level	S	N	N	
		Mult	I	N	N	
		State	I	N	N	
		Irrep	I	N	N	
MDCI_LED	N	numOfFragments	I	N	N	The number of fragments
		sumNonDisStrong	D	N	N	Sum of non dispersive correlation strong pairs
		sumNonDisWeak	D	N	N	Sum of non dispersive correlation weak pairs
		electrostRef	AD	N	N	Electrostatics reference
		exchangeRef	AD	N	N	Exchange reference
		dispContr	AD	N	N	Strong pair dispersion contribution
		dispWeak	AD	N	N	Weak pair dispersion contribution
		refInt	AD	N	N	Interaction reference
		corrInt	AD	N	N	Interaction correlation
		totInt	AD	N	N	Interaction total
Natural_Orbitals	N	NEL	I	N	N	
		NSOMO	I	N	N	
		NDOMO	I	N	N	
		NVMO	I	N	N	
		NNatoOrbs	I	N	N	
		OccUno	AD	N	N	Occupation of natural orbitals
		OccUnso	AD	N	N	Occupation of spin natural orbitals
Calculation_Timings	N	GTOINT	D	Y	N	
		STOINT	D	Y	N	
		SCF	D	Y	N	
		SCFGRAD	D	Y	N	
		GSTEP	D	Y	N	
		PLOT	D	Y	N	
		MOM	D	Y	N	
		MP2	D	Y	N	
		CPSCF	D	Y	N	
		LOC	D	Y	N	
		CIS	D	Y	N	
		MRCI	D	Y	N	
		REL	D	Y	N	
		SOC	D	Y	N	
		EPRNMR	D	Y	N	
		CASSCF	D	Y	N	
		MDCI	D	Y	N	
		CHELPG	D	Y	N	
		ROCIS	D	Y	N	
		MRCC	D	Y	N	
		FCI	D	Y	N	
		MD	D	Y	N	
		FREQ	D	Y	N	
		FREQ_NUM	D	Y	N	
		COMPOUND	D	Y	N	
		CIPSI	D	Y	N	
		AUTOCI	D	Y	N	
		ESD	D	Y	N	
		DLPNOCC	D	Y	N	
		MCRPA	D	Y	N	
		NEB	D	Y	N	
		CONFSC	D	Y	N	
		ANMR	D	Y	N	
		MM	D	Y	N	
		MMGRAD	D	Y	N	
		XTB	D	Y	N	

continues on next page

Table 9.1 – continued from previous page

		XTBGRAD	D	Y	N
		LFT	D	Y	N
		EXT	D	Y	N
		ONIOM_S	D	Y	N
		ONIOM_M	D	Y	N
		ONIOM_L	D	Y	N
		PROPINT	D	Y	N
		RESPONSE	D	Y	N
		PROP	D	Y	N
		CASRESP	D	Y	N
		MP2RESP	D	Y	N
		COSMORS	D	Y	N
		AUTOCIRESP	D	Y	N
		GOAT	D	Y	N
		CISRESP	D	Y	N
		EDA	D	Y	N
		SUM	D	Y	N
SCF_Timings	N	TOTAL	D	Y	N
		PREP	D	Y	N
		GUESS	D	Y	N
		FOCK	D	Y	N
		DENS	D	Y	N
		DIAG	D	Y	N
		ETOT	D	Y	N
		POP	D	Y	N
		TRAFO	D	Y	N
		ORTHO	D	Y	N
		DIIS	D	Y	N
		SOSCF	D	Y	N
		NR	D	Y	N
		J	D	Y	N
		X	D	Y	N
		XC	D	Y	N
		GRID	D	Y	N
		FSYMMETR	D	Y	N
		PSYMMETR	D	Y	N
		FOCKSTART	D	Y	N
		SOLV	D	Y	N
		SOLV_INIT	D	Y	N
		SOLV_FINAL	D	Y	N
		INT_PREP	D	Y	N
		INT_PRESCREEN	D	Y	N
		INT_BF	D	Y	N
		INT_DENS	D	Y	N
		INT_DENSIO	D	Y	N
		INT_FUNC	D	Y	N
		INT_POT	D	Y	N
		INT_POTIO	D	Y	N
		INT_SUM	D	Y	N
		SPLITRIJ	D	Y	N
		COSX	D	Y	N
		COSX1C	D	Y	N
		RIJK	D	Y	N
		STABANA	D	Y	N
		DFET	D	Y	N
		ADFT_GUESS	D	Y	N
		ADFT_JDENS	D	Y	N
		ADFT_JMAT	D	Y	N
		ADFT_XDENS	D	Y	N
		ADFT_XMAT	D	Y	N
		ADFT_TMAT	D	Y	N
		ADFT_EQNSOLV	D	Y	N
		ECRISM	D	Y	N
		ECRISM_QESP	D	Y	N
		ECRISM_TOOL	D	Y	N
		ECRISM_FOCK	D	Y	N
		FROZCORE	D	Y	N
Solvation_Details	N	Solvent	S	N	N
		Epsilon	D	N	N
		Refrac	D	N	N
		RSolv	D	N	N

continues on next page

Table 9.1 – continued from previous page

		SurfaceType	S	N	N		
		CPCMDielEnergy	D	N	N		
		NPoints	I	N	N		
		SurfaceArea	D	N	N		
Geometry	N	NAtoms	I	N	N		
		NCorelessECP	I	N	N		
		NGhostAtoms	I	N	N		
		Coordinates	CC	N	N		Bohr
		ATNO	AI	N	N		
		coreless_ECP	AI	Y	N	Non-zero for cECPs	
		ghostAtoms	AI	Y	N	Non-zero for ghost atoms	
		fragments	AI	Y	N	Fragment ID	

[1] The name in brackets (if given) is the base property name, which is used for JSON export.

[2] Are multiple properties with the same base property name allowed. If so, the JSON value is a list of properties.

[3] Type, abbreviated as follows: D: `Double`, C: `Complex`, I: `Integer`, B: `Boolean`, S: `String`, AD: `ArrayOfDoubles`, AC: `ArrayOfComplex`, AI: `ArrayOfIntegers`, AB: `ArrayOfBooleans`, CC: `Coordinates`,

[4] Whether the component is optional (only stored if applicable).

[5] Whether there are multiple entries of the component. In JSON, these will be stored as a list.

CHAPTER

TEN

ARCHITECTURE OF ORCA

10.1 ORCA Structure

10.1.1 Program Components

The program system ORCA consists of several separate programs that call each other during a run. The following basic modules are included in this release (listed in alphabetical order):

orca	Main input+driver program
orca_ailft	Ab Initio Ligand Field Theory
orca_autoci	CI type program using the automated generation environment (ORCA-AGE)
orca_casscf	Main program for CASSCF driver
orca_casscfresp	CASSCF static linear response
orca_cclib	Precalculation of one particle coupling coefficients for ACCCI
orca_chelpg	Calculation of different kinds of ESP charges
orca_ciprep	Preparation of data for MRCI calculations (frozen core matrices and the like)
orca_cipsi	Iterative Configuration Expansion Configuration Interaction (ICE-CI)
orca_cis	Excited states via CIS and TD-DFT
orca_cisresp	CIS response calculation
orca_compound	General driver of the ORCA compound language
orca_crystalprep	Set up an embedding calculation from a crystal structure file
orca_eca	Auxiliary program for solving magnetic model Hamiltonians
orca_eda	Energy decomposition analysis module
orca_esd	Excited state dynamics program for calculating vibronic spectra, resonance Raman, …
orca_grad	General gradient module
orca_guess	Generation of an intial guess for SCF and CASSCF
orca_leanscf	Self consistent field program for HF and DFT
orca_lft	Ligand Field Theory
orca_loc	Calculation of localized molecular orbitals
orca_mcrpa	CASSCF linear response for excited states
orca_md	Molecular dynamics program
orca_mdci	Matrix driven correlation program: CI, CEPA, CPF, QCISD, CCSD(T)
orca_mm	Classical forcefield components
orca_mp2	MP2 program (conventional, direct and RI)
orca_mrci	MRCI and MRPT calculations (individually selecting)
orca_ndoint	Calculates semiempirical integrals and gradients
orca_nmrspectrum	Simulates NMR spectra from calculated NMR parameters
orca_numfreq	Numerical Hessian computation
orca_pc	Addition of point charge terms to the one-electron matrix
orca_plot	Generation of orbital and density plots
orca_pop	External program for population analysis on a given density
orca_prop	Calculation of molecular properties
orca_propint	Calculation of property integrals

continues on next page

Table 10.1 – continued from previous page

orca_rel	(Quasi) Relativistic corrections
orca_rocis	Excited states via the ROCIS method
orca_scfgrad	Analytic derivatives of SCF energies (HF and DFT)
orca_scfresp	self consistent field response
orca_startup	Calculation of startup data before each single point
orca_vpot	Calculation of the electrostatic potential on a given molecular surface
orca_vpt2	VPT2 analysis

10.2 The Architecture of ORCA

In this chapter we describe the broad features of ORCA's internal structure. This knowledge is not necessary to carry out calculations with ORCA. It may, however, help to understand which modules are being called in which order and why this is happening in the sequence it does.

10.2.1 The structure of the ORCA source code

The source code of ORCA is broadly structured in three parts:

1) The main program.
2) The code for the "ORCA tool library".
3) The code for individual modules.

The code in the ORCA tool library is being compiled into one library file that subsequently is linked with all ORCA modules. The code for the individual modules can make use of everything that is in the library, but the modules are not supposed to link to or use any code of the main program or any other module. This way of structuring the code ensures that the modules remain maintainable and that there are no complex and unwanted interdependencies that would make it eventually impossible to exchange modules for new code.

10.2.2 The shell structure of ORCA

The whole organization of the code and the information flow can be thought of as consisting of a shell structure where a shell is a layer of software with well-defined functions and a well-defined interface to the shells above and below it.

The lowest layer of the shell consists of the SHARK integral package. This is the "motor" of the program that takes care of most compute intensive tasks and essentially all parts that involve integrals over basis functions. This amounts to calculating all one- and two-electron integrals, forming Fock and Fock-like matrices and performing integral transformations using all kinds of basis sets and kernels. Most of the SHARK code is independent of the remaining ORCA source code infrastructure.

The layer above SHARK is the actual ORCA source code. ORCA interacts with the user through the interface and it orchestrates the entire workflow of the calculation. At this point in time, it also performs compute intensive tasks like forming diagonalizing and handling Fock-operators , solving large linear equation systems, forming sigma vectors, densities etc in correlated calculations and similar tasks. The ORCA source code interacts with SHARK through a small set of DRIVERS. The drivers feature genuine ORCA data structures as well as SHARK infrastructure. The DRIVERS call the genuine SHARK functions (ORCA calls SHARK directly only in a few places) and, most importantly, the DRIVERS take care of finding their way through the approximation jungle. By this we mean they handle the necessary basis sets, auxiliary basis sets, grids, integral approximations, solvation, relativity, point charges etc.

Concentrating these important and highly error prone tasks in a number of well documented routines provides a highly transparent way of identifying and properly maintaining the functionality that is at the heart of the program.

Above the ORCA software layer there are tools that orchestrate workflows. Workflows typically consisting of a number of interdepend computational steps that are later combined into a single meaningful chemical result. For example, one may optimize the geometry with a DFT functional and calculate zero-point and thermodynamic corrections at the same level and follow it up with a single point calculation at the coupled cluster level that may or may not feature another correction for core correlation at the MP2 level or complete basis set extrapolation. Such tasks and many others, like running series of calculations on a test-sets of molecules or permuting calculation options like functionals or basis sets on a given test system can be addressed with workflow tools. Inside ORCA, there is the very powerful compound scripting language to achieve and organize such tasks. On the commercial branch, FAccTs develops the Weasel tool that is a powerful and highly reliable workflow organizer.

At the final layer, one might envision a graphical user interface (GUI) that helps building molecules and facilitates running calculations and analyzing results. At this point in time, ORCA does not have a dedicated GUI. There are many free and commercial solutions that interface to ORCA. This form of interfacing is facilitated by the `orca_2json` interface and the property file that ORCA produces. We hope that the transparency of this interface motivates GUI developers to provide ever improved GUIs that interface ORCA. We do not exclude the possibility that ORCA will feature it's own GUI sometime down the road.

10.2.3 The master/slave concept and the calling sequence

Within ORCA, the code follows a "master/slave" concept in which the main program is allowed to know everything about every module while the modules are not allowed to know anything about the main program. Thus, the main program is the piece of software that orchestrates the entire ORCA run and the interaction with the user. The main program is, however, not supposed to carry out compute intensive tasks or even parallelized tasks by itself.

An ORCA calculation commences with reading and analyzing the ORCA input file. The plausibility of this input is checked by an elaborate "maincheck" routine. Quite obviously, the number of possible combinations of ORCA features is far larger than what could possibly be checked with realistic effort. However, the maincheck routine has evolved over the year in a way that allows to detect the most common combinations of invalid or inconsistent input keywords and take appropriate action (that can be changing some parameters or aborting the job).

After through maincheck the basis sets used in the calculations and the coordinates are completely known. This is then the time to initialize the SHARK integral package and will then carry out the bulk of the computation heavy tasks.

The first module that is being called is `orca_startup`. This module takes care of calculating the one electron integrals such as the HCORE and overlap matrices, the metric matrices in RI calculations, the pre-screening matrices for direct SCF and possibly also the two-electron integrals for integral conventional runs.

The next module downstream is the `orca_guess` program. This module has the task to produce an initial set of molecular orbitals and also an initial density matrix from these orbitals. To this end, there are a number of options out of which the most common ones are to a model density guess (PMODEL) or to read orbitals from a previous

calculation (MOREAD). In the latter case, the calculation may feature a different geometry and/or a different basis set but the number, nature and ordering of atoms need to be consistent with the target system.

In the third step, the program branches out into either the SCF or the CASSCF module (`orca_leanscf` or `orca_casscf`). These modules produces converged orbitals and a self-consistent field energy as well as one-particle density matrix. The latter is stored in in the `DensityContainer`, which is a centralized storage facility for densities that will also be left over after the calculation and can be accessed by the users.

The next step of the calculation consists of calling various post-SCF programs like `orca_mp2`, `orca_mdci` or `orca_autoci`. NEVPT2 and CASPT2 are calculated by code that is integrated with `orca_casscf`. These calculate correlation energies and excited states among many other things.

There are many different pathways that the program can take next, for example geometry optimization or embedding calculations or molecular dynamics to only name a few. Rather than going into each and every one of the possible pathways, we will describe the calculation of molecular properties inside ORCA.

10.2.4 The calculation of molecular properties

For the calculation of molecular properties, ORCA has a fairly unique and strongly streamlined infrastructure that is focusing on the commonalities in the calculated properties and that are independent of the underlying electronic structure method used.

There are three qualitatively different sets of properties that are covered in ORCA property calculations:

1) Response properties. These are properties that can be formulated as derivatives of the total energy.
2) QDPT properties. These are properties that are calculated by quasi-degenerate perturbation theory (QDPT). This amount to diagonalizing the Hamiltonian containing external fields and relativistic corrections over a number of roots delivered by the underlying electronic structure method
3) Excited state properties: these are, at least at this point in time, transition moments between different electronic states of the system.

Response properties

A large number of properties can be written as derivatives of the total energy. These are first order properties:

$$X_M = \frac{\partial E}{\partial M} = \sum_{\mu\nu} P^{\pm}_{\mu\nu} \left\langle \mu \middle| \widehat{h}_M \middle| \nu \right\rangle$$

Here, $P^{\pm}_{\mu\nu}$ is the first-order electron ('+') or spin ('-') density $\widehat{h}_M$ is the operator that is describing the property of interest (e.g. the dipole moment operator) and the basis set is $\{\mu\}$.

Second order properties are:

$$X_{MN} = \frac{\partial^2 E}{\partial M \partial N} = \sum_{\mu\nu} P^{\pm}_{\mu\nu} \left\langle \mu \middle| \widehat{h}_{MN} \middle| \nu \right\rangle + \sum_{\mu\nu} \frac{\partial P^{\pm}_{\mu\nu}}{\partial N} \left\langle \mu \middle| \widehat{h}_M \middle| \nu \right\rangle$$

Here the important quantity is $\frac{\partial P^{\pm}_{\mu\nu}}{\partial N}$, the first derivative (or "response") of the electron or spin density with respect to perturbation N. It can be shown that the order of perturbations M and N is immaterial and hence one can choose the more convenient perturbation to calculate the response density for.

It is important to point out that the equations for response properties can *always* be brought into this form, irrespective of whether the calculation is a Hartree-Fock, DFT, MP2, coupled cluster, CASSCF or full-CI wavefunction.

Given the considerable generality of the property equations, it seems logical to create an infrastructure in which these similarities are exploited to the largest possible extent. In ORCA the central place where all densities and response densities are stored is the `DensityContainer`. This is used intensely throughout the actual calculation and left on disk as `BaseName.densities` where it can be used for visualization.

In terms of the calculation flow the main program contains all the logic to drive this calculation. It first drives the SCF and possibly the post-SCF calculations. After this is done, the SCF program collects the information about

which property integrals will be needed down the road and call the property integral program `orca_propint` that will calculate the necessary integrals and stores them in another central storage container, the "property integral container".

The next step of the calculation is that the main program determines for which properties response densities are needed. It collects these perturbations and calls the response modules. In the case of a SCF wavefunction (HF or DFT), this is the `orca_scfresp` program. This program will divide the requested perturbations into different types like, real-symmetric, imaginary-antisymmetric, triplet etc. Then it solves the response equations for all types of perturbations simultaneously. This leads to large efficiency gains since the same integrals are needed on the left-hand side of the coupled-perturbed SCF (CP-SCF) equations irrespective of the actual perturbations. Hence, pooling all perturbations of a given type together is much more efficient than treating these perturbations one at a time. This leads to high efficiency gains, in particular for properties like NMR parameters where nucleus dependent perturbations are required.

For a number of other electronic structure methods like MP2, CASSCF or AutoCI the respective modules can be run in "response mode" where instead of solving the equations for the energy, they pick up the amplitudes that were determined in the energy run and use them together with the property integrals in order to produce response densities.

The response densities are then stored in the density container too. At this point, one has everything that is required in order to calculate the actual properties – a task that is performed by `orca_prop`.

The `orca_prop` program works by browsing through the density container and looks for densities that are appropriate for calculating a requested property. As soon as it finds an appropriate density (or combination of unperturbed and response density), it will calculate the property. For example, if you have calculated SCF, MP2 and CCSD in the same calculation and have asked for the calculation of all three densities, `orca_prop` will calculate three dipole moments and print them right next to each other.

One beneficial side effect of this organization is that it is very well suited for future extensions. If a new method becomes available in ORCA that is able to produce densities and possibly response densities, the properties for this method will be fully available without any programmer needing to write a single line of additional code. It also ensures that all properties are calculated in a consistent fashion and with a consistent printout.

The results of all property calculations are stored in the central property file that will be left after the calculation. Users interested in reading properties, it should be read from the property file or its JSON translation.

QDPT properties

Some properties are not calculated as energy derivatives but from quasi degenerate perturbation theory. In this method one diagonalizes the QDPT matrix:

$$Q_{IJ} = \left\langle \Psi_I^{SM} \middle| E_I \delta_{IJ} + \widehat{H}_{SOC} + \widehat{H}_{SS} + \widehat{H}_{LB} + \ldots \middle| \Psi_J^{S'M'} \right\rangle$$

Here $\left|\Psi_I^{SM}\right\rangle$ are the roots of a given method that can generate excited states with energy E_I. For example, these can be TD-DFT roots, CASSCF roots, CASSCF roots with energy corrections from NEVPT2 or CASPT2, MRCI-roots, EOM-CCSD roots etc. and $\widehat{H}_{SOC}$ is the spin-orbit coupling (SOC) operator, $\widehat{H}_{SS}$ the electron-electron spin-spin coupling operator, $\widehat{H}_{LB}$ the molecular Zeeman operator etc. In practice, only the principle component $M = S$ (i. e. Ψ_I^{SS}) is calculated and the necessary matrix elements are generated using the Wigner-Eckart theorem.

The diagonalization produces the complex valued relativistic eigenstates as linear combinations of the non-relativistic or scalar relativistic eigenstates. Using the eigenstates relativistic densities or transition densities can be calculated that subsequently can be used to calculate magnetic properties or "relativistic" optical or magneto-optical spectra.

The procedure obviously suffers from a truncation error because only a finite number of roots can be calculated in practice. However, results indicate that the QDPT generated properties compare often very favorable with experimental data provided that the underlying electronic structure method delivers reasonable results.

In ORCA, all QDPT properties are calculated in a consistent infrastructure and then also leads to consistent printing and reporting of the results.

Excited state properties

Closely related to the QDPT infrastructure is the "one-photon spectroscopy" (*OPS*) infrastructure. This is an infrastructure that coordinates the calculation of transition moments using a set of transition densities as input. These might have been stored on disk or might have been generated on the fly. In a similar way to the QDPT infrastructure, these transition moments are generated in a consistent manner throughout all modules of ORCA that can generate excited states.

The exact field/matter coupling Hamiltonian is:

$$A(\mathbf{r}, t) = -A_0 \varepsilon \exp(i\mathbf{k}\mathbf{r} - \omega t)$$

Here $\mathbf{k}$ is the wavevector of the radiation with frequency ω and ε is its polarization. A_0 is the intensity of the radiation, $\mathbf{r}$ the electronic coordinate and t is the time.

Evaluation of the matrix element $\left\langle \Psi_I^{SM} \middle| \sum_i A(\mathbf{r}_i, t) \middle| \Psi_J^{S'M'} \right\rangle$ amount to generating the "exact" field-matter transition moments. This can be requested by setting the flag `DoFullSemiClassical = true` in the appropriate ORCA input block.

A series expansion of the cosine term yields the familiar electric dipole, electric quadrupole, magnetic dipole etc. contributions. Interestingly, the direct evaluation of the electric dipole term would yield it in the gauge independent "velocity" form. They are related to the more familiar "length" form dipole matrix elements by:

$$\left\langle \Psi_I^{SM} \middle| \sum_i \mathbf{p}_i \middle| \Psi_J^{S'M'} \right\rangle = \frac{-i}{\hbar}(E_J - E_I) \left\langle \Psi_I^{SM} \middle| \sum_i \mathbf{r}_i \middle| \Psi_J^{S'M'} \right\rangle$$

The results of the length and velocity transition moment evaluation are expected to match in the basis set limit if the electronic structure method at hand satisfies certain conditions. In practice, they usually do not agree very well, even if large basis sets are used.

In order to generate electric length and velocity as well as higher moment evaluations, you can use the keywords

```
DoDipoleLength   = true
DoDipoleVelocity = true
DecomposeFosc    = true
```

in the appropriate ORCA input blocks that trigger the respective excited state calculation.

10.3 Units and Conversion Factors

Internally the program uses atomic units. This means that the unit of energy is the Hartree (Eh) and the unit of length is the Bohr radius (a_0). The following conversion factors to other units are used:

1 Eh	=	27.2113834 eV	
1 eV	=	8065.54477 cm^{-1}	= 23.0605 $\frac{kcal}{mol}$
1 cm^{-1}	=	29979.2458 MHz	
1 a_0	=	0.5291772083 Å	
1 a.t.u.	=	2.4188843 10^{-17} s	

APPENDIX

A

DETAILED CHANGE LOG

A.1 Changes ORCA 6.1

A.1.1 New Features

Startup, Guess, LeanSCF

- BUPO algorithm.
- Significant memory improvements in Startup, Guess and LeanSCF.

DFT

- Added B3LYP-3c method.
- Extension of HF-3c, PBEh-3c, B97-3c, and B3LYP-3c for elements Fr-Pu (using original D3 parameters) and r2SCAN-3c up to Lr (Z=103).
- ADFT (Energies, gradient, response, TD-DFT, NOT GIAO, NOT analytic Hessian).
- RPA correlation energy a la Andreas Görling (no gradient, density or response yet; NumGrad, NumFreq).
- Simple input keyword for B2NC-PLYP double-hybrid.
- Added various simple input `LibXC(method)` keywords for DFT functionals, e.g. TASK-LDA, revTM, HSE06, GAS22.

Native xTB Methods

- Native implementation of GFN1-xTB with energy & gradient.
- Native implementation of GFN2-xTB with energy & gradient.
- Native xTB methods can use the ORCA SCF and plotting infrastructure. Plotting of molecular orbitals, densities, and fractional occupation densities (FODs) can be done with `orca_plot`.
- Spin-polarized methods spGFN1-xTB and spGFN2-xTB.
- JSON parameter file for custom parameterizations of GFN1- and GFN2-xTB.
- Native implementation of the ALPB solvation model for xTB methods.

Solvation

- Analytical DRACO gradient.

SCF Linear Response

- Analytical Raman intensities: HF, DFT up to meta-GGAs, RIJ and COSX, implicit solvation, external point charges. Significantly faster than NumFreq.
- Static dip/dip/dip hyperpolarizability: same range of validity as Raman intensities.

Orbital localization

- IAO-based valence virtual orbital localization methods (`PMVVO`, `LIVVO`).
- Different options for IAO minimal basis.
- Optionally use a different localization method for virtual MOs via `%loc`.
- LMO plotting enabled in input file.

Optimization

- New redesigned L-OPT - an L-BFGS optimizer for geometry optimization of large systems - more efficient than before.
- !RIGIDBODYOPT for optimization of structures/fragments as rigid bodies.
- Replaced frozen coordinates (`$` syntax) with X/Y/Z Cartesian constraints.

NEB

- More robust initial path generation using S-IDPP.

Excited states

- Core ionization/excitation energies using UHF IP-EOM-CCSD and STEOM-CCSD.
- Multi-core calculations for RHF/UHF Canonical and BT-PNO IP-EOM and STEOM-CCSD (DLPNO for now can only do 1 core at the time).

GS-ROCIS:

- Spin-orbit coupling is now implemented.

DeltaSCF

- Generalized mode following method.
- Freeze-and-release method.

MDCI

- Add `DT_in_Triples` correction for RHF reference with a check to skip for small F_{ia}.

AutoCI

- Third order NEVPT (NEVPT3).
- Fourth order NEVPT restricted to singles and double excitation (NEVPT4(SD)).
- Export single-reference CI/CC 2RDM to JSON.
- UHF CCSDTQ.

AutoCI Response

- RHF CCSD dipole-dipole polarizability, with and without orbital relaxation.
- RHF and UHF CCSD(T) dipole-dipole polarizability, with and without orbital relaxation.
- RHF and UHF CCSD quadrupole-quadrupole polarizability, with and without orbital relaxation.
- RHF and UHF CCSD(T) quadrupole-quadrupole polarizability, with and without orbital relaxation.
- RHF and UHF GIAO-CCSD NMR shielding, with and without orbital relaxation.
- RHF and UHF GIAO-CCSD(T) NMR shielding, with and without orbital relaxtion.
- UHF GIAO-CCSD EPR g-Tensor, with and without orbital relaxation.
- UHF GIAO-CCSD(T) EPR g-Tensor, with and without orbital relaxation.

CASSCF Module

- ICE NEVPT2 using the full ranked formulation (ICE-FR-NEVPT2).
- Multiconfiguration pair density functional theory (MCPDFT)
  - with translated functionals tLDA, tPBE, and tPBE0
  - and with complex translated functionals ctLDA, ctPBE, and ctPBE0
- Long-range CASSCF short-range DFT (srDFT)
  - with short-range functionals sr-LDA, sr-PBE, and sr-PBE0
  - short-range pair density functionals: sr-ctLDA, sr-ctPBE, and sr-ctPBE0
  - robust RI approximation for range-separated integrals used for RI-J and RI integral transformation
- Effective Hamiltonian CI-DFT approach for excited states with MCPDFT and srDFT.

CASSCF Linear Response

- New options to toggle the orbital response, solution locking, and Olsen preconditioner in `%casresp` block.
- Nuclear-orbit perturbation implemented (allows for calculation of `Aorb` contribution to EPR A-tensor).
- Block preconditioner to reduce the number of iterations and computation time.

MR methods

- RASCI / ORMAS module.

MCRPA module

- Block preconditioner to reduce the number of iterations and computation time.
- Orbital response can be turned off. This corresponds to a CAS-CI calculation.

Multiscale

- Automated generation of QM and active regions.
- MM-only calculations can now make use of active atoms keyword.
- Added fast Simple_Eq option for topology preparation of QM/QM2 runs.

MD Module

- Langevin thermostats are now available (random force and drag term can be switched on separately).
- Constraints: Can now automatically constraint all bonds/angles involving a certain element (e.g. H).
- Restraints: Can now define custom restraint function by arbitrary mathematical expression.
- Added new "Radius of Gyration" Colvar (can restrain the system to spherical shape, e.g. a drop).
- Metadynamics: Can now define custom Colvars by arbitrary mathematical expression.
- Can now modify atom masses during MD simulation.
- Many more options for velocity initialization (e.g. let certain molecules rotate or translate).
- "Reference trajectory" feature to re-compute snapshots from an existing trajectory. Can also compute work along a path.
- Wavefunction extrapolation (ASPC) to reduce number of required SCF cycles.
- Possibility to apply additional arbitrary forces to atoms (e.g. for experiments with own bias potentials).

Thermochemistry

- Diagonal Born-Oppenheimer correction.
- Spin orbit coupling correction to the total energy via approximate 2 components.

NMR

- Enabled shielding calculations with X2C and the finite nucleus model.

Energy Decomposition and Wavefunction Analysis

- Atomic decomposition of London dispersion (ADLD).
- Atomic decomposition of exchange energy (ADEX).
- LED for covalent bonds (COVALED).
- New implementation of ETS-NOCV.
- Enabled LED with X2C Hamiltonian.
- RIJCOSX can now be used for DLPNO-CCSD(T)-LED with non-hybrid DFT as reference wavefunction.

AILFT

- Extended Spaces AILFT (ESAILFT).

Magnetic Relaxation Rates

- Orbach and Raman II pathways (`orca_magrelax`).

RESP/CHELPG charges

- RESP charges with parabolic and hyperbolic constraints.
- Can now define sets of equivalent atoms which will have identical charges.

ANISO

- Update of the ANISO interface in ORCA to support MRCI.

Basis Sets

- 6-31G* for Ga-Kr.
- CRENBL basis and CRENBL-ECP.
- New Jensen basis sets: pcX-n, aug-pcX-n, pcH-n, aug-pcH-n (n = 1,2,3,4).
- Reimplemented def-SVP, def-SV(P), def-TZVP, def-TZVPP (H-Lr).
- Extended def2-mSVP, def2-mTZVP, and MINIX up to Lr.
- Extended def2/J option for elements Fr-Lr.
- Extended def2/JK option for elements La-Lu, Fr-Lr.

Property File

- Added SCF and job timings.
- Added DBOC, gCP, and SOC energy contributions.
- Added ESD fluorescence rates.
- Easier conversion to JSON via `%output JSONPropFile=true`.

orca_2json

- Compute and export AO-basis Fock matrix as well as separate J, K, XC, and CPCM terms.
- Convert JSON property files back to native ORCA format (with limited support for 6.0 files).
- Added SOC integrals.

Miscellaneous

- Geometry input from GBW files.
- TransInvar for Numerical Hessian.
- Thermochemistry Compound script for the HEAT protocol by Stanton et al.

A.1.2 Improvements

TDDFT

- Improved `FollowIRoot` stability.
- TD-DFT amplitudes now stored in GBW file.
- TD-DFT calculations are restartable from a GBW file.

GOAT

- GOAT-COARSE option for automated coarse-grained conformational search.
- GOAT-DIVERSITY for a conformational search looking for maximal structure diversity.

DOCKER/SOLVATOR

- Faster by about 100x, with many improvements and updates.
- DOCKER can now do the search only on limited region of the host by defining a center and extent of the search.

AutoCI

- Significantly improved scaling and runtime performance (speed, memory and disk space requirements) for AUTOCI-MP5 energy.
- Improved RHF CCSD(T) 2RDM performance.
- UHF CCSD(T) gradient accuracy (disconnected 3rd and 4th order contributions were missing, now corrected).
- Response: UHF CCSD Polarizability, solver performance improved significantly.
- Response: switched to sequential handling of response equations (less disk requirements for large calculations, compared to solving all perturbations together).
- Printing improvements.

CASSCF

- Aborts at beginning if CASSCF is requested with a dispersion correction.
- General warnings added about number of roots when CASSCF LR is run.
- TRAH submodule: final active orbitals are the same and sorted the same as in other CAS convergers.

CASSCF Linear Response

- More reasonable threshold for LR equation solution is determined by default.
- Uses solution locking by default.
- Checking at start that there are response parameters with which to perform LR calculation.

OpenCOSMO-RS

- Gas-phase orbitals can be used for the calculations in solution.

MD Module

- Added many more warnings (e.g. when two atoms come too close during a simulation).
- Added many different options how to handle non-converged SCF.
- Random velocity initialization now yields a zero angular momentum by default (i.e. system will not rotate).

Thermochemistry

- Extended max number of allowed temperatures and pressures to 100.

DKH

- Only compute the shell-pair data once.

X2C

- Skip grid setup if not needed.

OPS refactor

- Important: keywords to request multipolar spectrum has been simplified. Some Orca 6.0 keywords are no longer available.

QDPT module

- g-tensor `g(OZ/SOC)` now includes QED correction factor.
- g-tensor `ggauge` and `g(OZ/SOC)` can now be printed in molecular frame (with higher print level associated with QDPT).
- Enabled sum-over-states for CAS and MRCI.
- Reduced memory usage for SSC ZFS calculation.

Property File

- All properties and components now documented in the manual.
- Some components are now stored optionally/conditionally.
- Disable property file storage by default for `!MM` and `!GFN-FF` (in addition to `!GOAT` and `!MD-L-Opt`).
- Common component structure for energy properties:. `SCF_Energy`, `AutoCI_Energies`, `CAS_SCF_Energies`, `CAS_DCD_Energies`, `CAS_PT2_Energies`, `CIS_Energies`, `MDCI_Energies`, `MP2_Energies`.
- `Calculation_info`: Removed (unassigned) `NumOfFCElectrons` and `NumOfCorrElectrons`; moved "final single point energy" to geometry-dependent `Single_Point_Data` property.
- Stored MR-AUTOCI energies.
- Method-specific population analyses.
- Method- and state-specific nuclear gradient, removed `ATNO` component.
- `G_Tensor`: also store `g_PSO`, `g_DSO`, `Delta_g` components.
- `A_Tensor`, `EFG_Tensor`: removed `activeNucs`.

Miscellaneous

- Default MaxCore increased to 4GB.
- UCO/UNO generation performance improvement.
- Allowed simple input of DSD-PBEP86, DSD-BLYP and DSD-PBEB95 with D4.
- DeltaSCF: improved preconditioner with SOSCF.
- NEB: fragment preparation.
- Various improvements in `orca_mapspc`.
- Various fixes, additions, and improvements in Compound.

A.1.3 Fixed

TDDFT

- DCORR should now respect MaxCore and abort neatly if it is insufficient.
- Wrong JSON output of TDDFT/TDA amplitudes and TDA excitation energy.
- Fix bug of TDDFT calculations with semiempirical methods.
- Fixed GS energy storage when SOC is enabled.

MDCI

- Respect MaxCore when building CPCM containers for PTE(S) and PTES.
- UHF-based CCSD producing wrong results with `!CONV`.
- Race condition resulting in wrong results in parallel for PTE(S) and PTES.
- S-diagnostic for canonical RHF and UHF.
- Fixed integral transformation problems.
- Fixed problems with Brückner and orbital optimized UHF CCD.
- Estimate required memory in RI/DLPNO exchange-like integral half-transformation to prevent OOM death.
- Stop calculation if MDCI equations do not converge (same as for SCF).
- Rare pair-pair integral crash.

MP2

- MP2 with ANOs works now.

CASSCF/NEVPT2

- FIX: Custom `EDIAG` failed due to wrong unit conversion - the input energies should now be in Eh.
- FIX: Increased number of states for custom EDIAG (hard-wired).
- FIX: Abort if an unsupported CI solver is requested with TRAH.
- CABSSinglesCorrection is now default in a DLPNO-F12-NEVPT2 (DoCABSSingles=1).
- FIX: MaxIter not properly passed in DLPNO-NEVPT2.
- FIX: QD-NEVPT2 densities not properly used in property calculations.
- FIX: NEVPT2 property calculations now track ground state changes.

MCRPA module

- Removed runtime bottleneck in final CI wave function printing that became an issue for many roots and long CI expansions.

QDPT Module

- `ggauge` was still using `CenterOfNucCharge` for magnetic gauge origin; now uses globally set magnetic origin.
- Other g-tensor fixes to first-order contributions (now consistent with `orca_prop`).
- Issues with QDPT transition densities from ROCIS.

CASSCF QDPT properties

- Corrected indexing of transition densities and ground-state spin densities.
  - Affected properties: EFG tensor, RHO, first-order g-tensor contributions, SOS properties.
- Memory issue in SSC calculation.

QDPT Module Sum-over-states

- Runs now if g-tensor is requested with `DoSOS true` (before, A-tensor had to also be requested).
- SOS g-tensor now gives full agreement with CAS LR (without orbital response) when all roots are included in CAS/MRCI QDPT calculation.
- Corrected general retrieval of transition spin densities.
  - NOTE: SOS is thus only available for methods where transition spin densities are available (now: MRCI, CASSCF, CASSCF+NEVPT2, CASSCF+Custom diagonal energies)

MD Module

- External forces on Colvars are now properly computed (relevant e.g. for thermodynamic integration).
- Many bugfixes (e.g. for the "too many arguments" bug).
- Gradients were not computed for CIS and broken symmetry.

Solvation

- Various options for CPCM: analytic, numeric RI, multipole.
- DRACO keyword not ignored anymore for QM/MM calculations.
- Fix crash for calculations requesting CPCM (bare CPCM / SMD / openCOSMO-RS) if the atomic number of an element in the solute is >= 106.

Dispersion

- Additional `!ATM` alias for the Axilrod-Teller-Muto type three-body-dispersion treatment available via `!ABC`.

EPR

- Crash in `orca_euler` when number of EFG and HFC nuclei differ.

NMR

- ZORA NMR now automatically triggers numeric GIAO 1-el integrals, as in ORCA 5.
- Fixed parallel race condition in DLU-X2C NMR leading to wrong results.
- `Tau=Dobson` default now correctly applied to GIAO calculations.
- Abort for NMR couplings with ZORA (not implemented).
- Fixed hanging of DSO coupling term with X2C when shieldings are also requested.
- Fixed GIAO XC integration for B97-3c with XCFun.

Optimization

- The "big step" fallback should also work if only Cartesian constraints are used.
- Switch off group parallelization if `(Re)CalcHess` is needed.

Relaxed Surface Scan

- Now it allows negative dihedrals on input, improved error report.

DOCKER/SOLVATOR

- Issue when swarming the best structures.
- Fix the chirality of the moment of inertia to avoid creating stereoisomers.
- Wall potential issue when some atoms are very far from the centroid.
- Small fix for solvent dimensions.

Hessian

- Fix for NearIR in `PrintThermoChem` if `InHessName` is different from `BaseName.hess`.
- Fixed issue when writing the IR spectrum multiple times.

X2C

- Disable general contraction for decontracted ANO basis, which led to slowdown.
- CIS/TDDFT gradients with X2C were not picture-change corrected.

Thermochemistry

- D(inf)h point group was not correctly parsed for rotational entropy contribution.

Inertia Tensor

- VPT2 and PAF : Fixed Inertia Tensor calculation.

Rotational Constants

- Minor fixed on unit transformation constants.

Basis sets

- All of SHARK and ORCA now works up to L=10 (except Hessian).

SCF Linear Response

- Fixed issue with initialization of BHP22 solver.

Windows

- Windows-specific crash in `orca_mapspc` fixed.
- Parallel subcalculations in NumCalc (NEB, NumGrad, VPT2, …) now also available on Windows

Property File

- Re-implemented JSON output via the N. Lohmann library (version 3.11.3) to guarantee valid syntax.
- Alleviated memory and timing overhead.
- Respect `%output PropFile=false`/`!NoPropFile` keywords.
- `Quadrupole_Moment`: `EXZ` and `NUCXZ` components held YZ value instead.
- Hessian was stored incorrectly in XTB jobs.
- CIS/TDDFT gradient root was stored incorrectly.
- `Chemical_Shift` components `SPSO`, `SDSO`, and `saniso` were not stored.
- `Calculation_Info` was not stored in some calculations.
- MM, XTB, and ExtOpt jobs now store the geometry and final energy.
- Geometry in JSON output was missing ghost, fragment, and ECP data.

Miscellaneous

- Crash in SOSCF.
- Cleanup leftover files of stability analysis with DFT.
- Atomic fitting density not working with partial general contraction.
- Occasional crash in integral loop.
- Missing PDB file output in QM/MM NEB.
- Prevent crash when reading the nodefile in GOAT, NumFreq, etc.
- Bug in one-sided numerical gradient `TransInvar=true`.
- Restored functionality of `!AIM`.
- Removed leftover temporary files in AUTOCI.
- Restore original irreps if the prediagonalization is skipped in SCF.
- Crash in Hessian for systems with no beta electrons.
- Restored functionality of the `!DoEQ` keyword.
- Opt+NumFreq or Opt+IRC jobs with `!ExtOpt` were stopping after the optimization.
- Print headers for all monitored internals in IRC.
- Fixed HFLD when no pairs are identified.
- Removed RIJCOSX-only restriction for wB97X-3c.
- CP-SCF now aborts if unconverged.
- RI Trafo in CIPSI works now.

- Fixed memory leaks in AO-MP2, QDPT, `orca_2json`, ECPs, ROCIS, LFT, and others.
- Spectrum intensity printing and decomposition in `orca_mapspc` for cases beyond the dipole approximation.
- `orca_vpot`: Fixed ESP calculation when contraction change.

A.1.4 Removed

CASSCF/NEVPT2

- QD-NEVPT2 with Cloiszeaux Hamiltonian (QDType=QD_Cloiszeaux).
- Some options for the handling of reduced density matrices in NEVPT2 (D4Step=core, D4Step=D4PT).

MCRPA module

- If the eigenvalue equations fail to converge, the connection to CASSCF wave function instability is not mentioned anymore. The manual section has also been removed.

A.2 Changes ORCA 6.0.1

A.2.1 Fixed

DFT

- Crash when XCFun functional was overwritten with LibXC.
- Hessian fixed for DFT GGA, NoRI, RKS.
- VV10 Hessian is blocked even if invoked with `CALC_HESS=TRUE` and similar.
- Wrong orbitals for non-self-consistent DFT-NL calculations (wB97M-V, wB97X-V, B97M-V).
- `X_WR2SCAN`:
  - Exchange can now be specified individually in the `%method` block.
  - Fixed crashes when second derivatives are requested.
  - TRAH is now disabled per default when using `X_WR2SCAN`.
  - If second derivatives are requested, will now automatically switch to numerical second derivatives.
  - Added appropriate warnings for the above changes.
- Fixed PBEh-3c gCP parameters for Krypton and Lithium to be consistent with Grimme's stand-alone.
- gCP is now fixed (and extended) for r2SCAN-3c up to Z=103.
- Remove restriction to COSX for wB97X-3c .

TD-DFT

- Ground state gradient for TDDFT calculations with `sgradlist` was wrong.
- `DCORR` 2/3 with `DoSCS` giving wrong results in parallel.
- (D)-Correction not available for full TDDFT.
- Fix for memory estimates for Hessian/TDDFT when running without COSX.
- `FollowIRoot` was not supposed to do anything if the overlap was to small, was still updating.
- Fixed interface to `BHP22` solver in CIS.

MP2

- Crash in conventional U-MP2.
- Parallel crash in RI-MP2 density.
- Crash in (RI-)MP2 gradient with SMD.
- Crash in (RI-)MP2 density with PGC and RIJK.
- MP2+CPCM gradient was wrong.
- Crash in NearIR + B2PLYP.
- Bug with MP2 gradient in property file.

MDCI

- SemiCore was not applied correctly if ECP is present.
- ECP-related crashes.
- Fixed redundant integral generation for specific problems.
- Restored old CITrafos to address reported performance issues.
- Added missing 4th-order doubles term in (T) for RKS reference (already present in UKS-(T), RKS-DLPNO-(T), and UKS-DLPNO-(T) and zero for RHF/UHF reference).
- UHF CIS/STEOM calculation with `UseCISUpdate` is set to `false`.
- RHF STEOM: TD-DFT initial guess.

AutoCI

- Fixed large stack allocation, e.g., in MRCC.
- Fixed runtime behavior for `!Moread Noiter` (falsely reporting "not converged").
- Fixed `!UseSym` falsely aborting.
- AutoCI gradients: abort at start of a calculation when RI is requested instead of after coupled cluster iterations.

CASSCF/NEVPT2/QD-NEVPT2

- Issue running LR over SA-CASSCF solution.
- Incorrect setting of gauge origin in CASSCF QDPT led to misleading output and in some cases complained about not being able to find densities for the origin evaluation.
- Canonicalize the inactive and virtual spaces of AVAS guesses, to avoid spurious warnings about core orbitals in the following CASSCF calculation.
- AVAS: fixed wrong number of occupied orbitals in case no occupied orbitals should have been selected.
- TRAH-CASSCF: compute generalized Fock matrix which is needed for the CASSCF nuclear gradient.
- Fixed redundant generation of coupling coefficients in the CI guess.
- Fixed ABS/CD spectra in calculation with !UseSym and QD-NEVPT2: The wrong densities were picked for the CASSCF transition moments.
- Fixed ABS/CD when the NEVPT/QD-NEVPT2 ground state differs from CASSCF. Respecitve transition were missing.
- Fixed MCD spectra not using transition densities from QD-NEVPT2 for the flag `DoFullSemiclassical=true`.
- Updated manual: Reported `D4TPre` are updated to the new default value 1e-12. ORCA 5 used `D4TPre=1e-10`.
- Fixed closed-shell case, e.g. CAS(6,3), crashing in NEVPT2.
- Fixed NEVPT2/FIC-NEVPT2 wrong energies or crashing for the Vija class to wrong addressing.
- Fixed ICE densities not stored in density container.

ANISO

- Fixed T and L matrices passed to the single-aniso.
- Fixed wrong number of non-relativistic states passed to single-aniso.

QDPT

- Corrected QDPT transition density for excitations beyond “none”.
- Information added to QDPT AMatrix.
- Issues in QDPT properties in `orca_lft` have been addressed.

QM/MM

- Speed issue for QMMM optimizations.
- Crystal-QMMM and compound crashed.
- Removed leftover files from QMMM-IRC amd QMMM-NEB.

Relativity

- Crash for F12 + X2C/ZORA/DKH.
- Unnecessary abort in AutoCI gradients with X2C/DKH/ZORA.
- Disabled X2C+GIAO+FiniteNuc (not yet implemented).
- 2nd-order PC correction to DKH gDSO now skipped when `fpFWtrafo==false` due to numerical instability.

Solvation

- Disabled analytical gradient and Hessian for XTB calculations requesting CPCMX (not implemented).
- FINAL SINGLE POINT ENERGY for calculations requesting CPCMX was wrong.
- Crash for calculations requesting Freq + CPCM + dummy atoms.
- Crash for QM/QM2 calculations with CPCM requesting excited states.
- Crash for multiple XYZ File Scans for DRACO.
- Crash with CPCM + NoIter + Pal + open-shell.
- Fix for GC and CPCM.

Optimization

- Multi-XYZ optimization crash.
- Random possible break when using GFN-xTB Hessian.
- Random crashes for `RECALC_HESS=TRUE`.
- COPT was saving wrong Cartesian Hessian under certain conditions, would break.
- Maximum number of angles that can be included is fixed + better error message.
- Analytic Hessian as initial Hessian option crashed with IRC.
- Crash in NEB-TS with subsequent Hessian, caused by change of number of parallel processes for NEB (max 32).
- NEB parallelization (will - again - automatically start in parallel, if enough processes are available).

GOAT

- GOAT/DOCKER/SOLVATOR now running on Windows.
- WorkerRandomStart fixed and working as intended.
- `-REACT` and `-EXPLORE` were (by mistake) not included sqrt(NFrag) to number of opts.
- Missing timings for GOAT.

DOCKER

- Abort if all final optimizations fail, was ending normally.
- Do not switch to COPT if constraints are given.

Stability analysis

- Stability analysis + closed-shell systems + post-processing (Hirshfeld, NBO, …).
- `SkipSecondSTAB` was still checking for energy differences between steps. Now will move on regardless.

`orca_2json`

- Exported relativistic integrals were wrong in, HMO and angular momentum were missing.
- Choice of origin corrected.
- Empty `[]` and `[""]` are no longer crashing but disabling the options.
- Invalid property JSON syntax in the following cases:
  - multiple geometries (e.g. optimizations);
  - some jobs with multiple properties of the same kind;
  - CIPSI energies;
  - MDCI EOM energies;
  - XTB jobs;
  - energy extrapolation.

`orca_mapspc`

- XAS/XES broadening functions satisfy FWHM.
- Adjusted .stk files normalization to report band integrals.

Compound

- MORead with same type and number of atoms but different arrangement.
- Bugs in statistical functions.

Miscellaneous

- SOMF(1X) parallel bug in semi-numeric Coulomb.
- Dummy/ghost atoms lead to crash in Hessian (partial fix).
- Fixed bug of Fermi smearing calculations of two-electron systems.
- Hangup in `leanscf_aftermath` when using F12 and ECPs.
- Issues in RIXSSOC, XESSOC spectra in ROCIS have been addressed.
- Issues in computing RI-SSC Integrals have been addressed. This property is now turned on in CASSCF, LFT and MRCI modules.
- Fixed a crash in MD and L-OPT when the input file name was “orca”.
- Default COSX algorithm is set to AUTO everywhere, as originally intended.

- Disable frozen-core approximation when no frozen-core electrons are present.
- `orca_vib` was not able to read hess file from AnFreq run.
- For very small systems restart Hessian could crash.
- Small deviations between the Guess CI Matrix and the Sigma Vector in GS-ROCIS.
- DCD-CAS: Removed left-over files.
- Uncontracted MRCI: Fixed partial general contraction calls in the MRCI integral transformation (crashed before).
- Crash for geometry optimization followed by a vibrational frequency calculation with fixed point group Ci.
- Removal of `posix_memalign`, due to glibc/kernel bug.
- NBO communication fixed.
- Fixes a crash in the integral transformation.
- Crash in `orca_vpot` due to missing prescreening matrix.

A.2.2 Improvements

Output

- Print all orbital energies for `!PrintMOs` and `!LargePrint`.
- Removed redundant warning when using gCP for elements Z > 36 (Kr).
- Added citations for wr2SCAN and DFT-D4 extension.
- Better printing of the spin coupling situation of the states resulting from GS-ROCIS calculations.
- Prepended a counter to irrep labels when printing vibrational frequencies.

`orca_2json`

- Citations added to json output file.
- Absolute path in basename possible.

`orca_mapspc`

- Added support for VCD, XASSOCV and XESSOCV spectrum processing.

Symmetry

- Ensured correctness of gradient cleanup, geometry optimizations with fixed point groups and calculations of vibrational frequencies (for point groups with real irreps using pure Hartree-Fock).
- Ensured correctness of the petite-list algorithm for SCF energy and gradient.

Compound

- Implemented automatic knowledge of basenames.
- Added GOAT interface.

Miscellaneous

- Added ASCII checker to input file.
- Add the possibility to read multi-XYZ files with no '>'.
- QDPT in CASSCF now uses the magnetic origin as defined in `%eprnmr` (if not set, defaults to CenterOfNucCharge for backwards compatibility).
- Reduced disk usage and optimized performance for CASSCF (transition) densities in density container.
- Keep topology in initial IDPP path generation.
- Add CIS Gradient in property file.

A.3 Changes ORCA 6.0.0

A.3.1 SCF and Infrastructure

- Significant improvements to the SOSCF solver to make it more robust, preventing huge steps that break the SCF. Overall improvements on the DIIS solvers.
- Due to the SCF updates, the AutoTRAH is now not so often needed and will start now only from above 50 cycles (`AutoTRAHIter`).
- Improvements to the memory handling of TD-DFT, CP-SCF and the Hessian

A.3.2 Basis sets

- def-TZVP and ma-def-TZVP pseudo-potential basis sets for the actinides (Z = 89, Ac - 103, Lr)
- Lehtola's hydrogenic gaussian basis set family (HGBS) including polarized (HGBSP) and augmented (AHGBS, AHGBSP) variants for all elements up to Oganesson (Z = 118)
- def2-SVPD, def2-TZVPD, def2-TZVPPD, def2-QZVPD, def2-QZVPPD basis sets for lanthanoids
- vDZP Grimme's double-zeta valence basis set
- `!MINIX` now correctly activates the corresponding ECP
- Added user-specified L-limit to AutoAux `AutoAuxLLimit`
- Fixed segfault in dhf-ECP
- Fix for `DelECP` in `%coords`
- Added `ReadFragBasis` keywords read fragment-specific basis sets from a file

A.3.3 Solvation

- New charge correction / compensation algorithm (corrected charges printed in an additional file)
- C-PCM/B scheme for QM/MM calculations
- DDCOSMO and CPCM/X available for XTB calculations and QM/MM calculations
- Generalization of names within all solvation models (C-PCM/SMD/ALPB/DDCOSMO/CPCM-X)
- New discretization scheme for the cavity (C-PCM) based on a constant number of charges per unit of area

A.3.4 DFT

- Allow LibXC functional customization via external parameters
- Simple input keywords added for some LibXC functionals
- Added wB97M(2) functional parameters: must be used with wB97M-V orbitals in a two-step job (compound script available)
- D4 for elements 87 (Fr) - 103 (Lr)
- r2SCAN-3c extension to elements 87 (Fr) - 103 (Lr)
- Simple input keyword for functionals with revised D4 parameters by Grimme (wB97X-D4rev, wB97M-D4rev)
- New hybrid functionals: r2SCANh, r2SCAN0, r2SCAN50, wr2SCAN, wB97X-3c
- New double-hybrid functionals: Pr2SCAN50, Pr2SCAN69, wPr2SCAN50, kPr2SCAN50
- Simple input keywords for 2021 variants of revDSD-PBEP86-D4 and revDOD-PBEP86-D4
- Bugfixes for LibXC combined `*_xc_*` functionals
- Fixed crash for D4 + ghost atoms

A.3.5 Excited states

- Analytical gradient for meta-GGA functionals
- Small bugfix to spin-adapted triplets and NACMEs.
- The FolllowIRoot for excited state optimization uses now a much more robust algorithm.
- New implementation for UHF EOM-CCSD and STEOM-CCSD.
- Core ionization/excitation energies using UHF IP-EOM-CCSD and STEOM-CCSD.

A.3.6 Relativity

- Enabled `NumGrad` with relativistic methods
- Second order DKH picture-change correction of contact density
- Minor fixes in DKH picture-change corrections of magnetic properties
- Picture change corrections are activated automatically

A.3.7 Multiscale

- Reading PDB files for 10k+ atoms with HETATMs now possible
- Enabled correct FlipSpin behavior with QMMM
- More efficient MM Module
- Implemented wall potential

A.3.8 Coupled cluster / DLPNO

- Implemented energy ordering for PNO generation
- Added semicore treatment for DLPNO
- Enable DLPNO-CCSD(T) calculations to run DLPNO-CCSD unrelaxed densities

A.3.9 MP2

- Corrected memory estimates and batching in response and gradient
- Removed the slow and limited analytic (RI-)MP2 Hessian code
- Removed non-default Gamma-in-core option for RI-MP2 response
- Disabled single-precision calculations
- Disabled SemiDirect option in AO-MP2
- Enabled range-separated DHDFT gradients with RIJDX

A.3.10 NEB

- Improved IDPP initial path
- More efficient GFN-xTB runs for NEB

A.3.11 COSX

- Improvements to numerical integration grids, both for DFT and COSX
- Faster grid step
- Improved performance and accuracy in COSX, also for the gradient and Hessian

A.3.12 Properties

- NMR spin-spin coupling:
  - Added `SpinSpinElemPairs` and `SpinSpinAtomPairs` keywords to limit which couplings are computed
  - Reduced the number of CP-SCF perturbations necessary via a stochastic selection
  - DSO term was transposed.
  - Off-diagonal PSO elements had the wrong sign
  - Efficiency improvement: solve SD/FC CP-SCF equations in restricted mode for RHF, instead of always using UHF
- Optimized numeric integration for HFC gauge correction

- Removed `RITRAFO` option for CP-SCF
- Switched to `tau=Dobson` as default handling of the kinetic energy density in meta-GGA magnetic properties with GIAOs

A.3.13 Hessian

- Improvements to the Hessian to avoid accumulation on numerical noise and reduce the number of spurious negative frequencies.

A.3.14 Geometry Optimization

- Several improvements to the geometry optimization, making is much more stable. Complete redesign of the Cartesian optimizer (!COPT), making it quick enough to be used together with faster methods.
- Fallbacks in the geometry optimization in case something fails, e.g. if the internal coordinates are unacceptable.
- Arbitrary spherical, ellipsoidal or box-like wall potentials can be added, which will reflect on the energy and gradients and can be used during geometry optimization.

A.3.15 Miscellaneous

- CHELPG charges that reproduce the ESP together with the molecular dipole moment
- Fixed issues with constraints in multi-step jobs
- Molden output: store ECP info in `[Pseudo]` block, set point charge atomic number to 0, handling of ghost atoms
- Made the `ExtOpt` interface easier to use
- Store energy from NEB and IRC in the XYZ file

APPENDIX

B

PUBLICATIONS RELATED TO ORCA

The generic references for ORCA are:

- Neese, Frank. The ORCA program system. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2012**, 2 (1), 73–78. DOI: http://doi.wiley.com/10.1002/wcms.81.
- Neese, Frank. Software update: the ORCA program system, version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2018**, 8 (1), e1327. DOI: http://doi.wiley.com/10.1002/wcms.1327.
- Neese, Frank. Software update: The ORCA program system—Version 5.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2022**, 12 (5), e1606. DOI: 10.1002/wcms.1606.
- Neese, Frank. Software Update: The ORCA Program System—Version 6.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2025**, 15 (2), e70019. DOI: 10.1002/wcms.70019.
- Neese, Frank; Wennmohs, Frank; Becker, Ute; Riplinger, Christoph. The ORCA quantum chemistry program package. *J. Chem. Phys.*, **2020**, 152 (22), 224108. DOI: https://aip.scitation.org/doi/10.1063/5.0004608.

Please do not only cite the above generic reference, but also cite in addition the original papers that report the development and implementation of the methods you have used in your studies! The following publications describe functionality implemented in . We would highly appreciate if you cite them when you use the program.

B.1 Method development

B.1.1 2025

1. Wittmann, Lukas; Garcia-Ratés, Miquel; Riplinger, Christoph. Analytical First Derivatives of the SCF Energy for the Conductor-like Polarizable Continuum Model with Non-Static Radii. J. Comput. Chem., **2025**. DOI: 10.1002/jcc.70099.
2. Müller, Simon; Nevolianis, Thomas; Garcia-Ratés, Miquel; Riplinger, Christoph; Leonhard, Kai; Smirnova, Irina. Predicting solvation free energies for neutral molecules in any solvent with openCOSMO-RS. *Fluid Phase Equilibria*, **2025**, 589, 114250. DOI: 10.1016/j.fluid.2024.114250.
3. Kempfer, Emily M.; Sivalingam, Kantharuban; Neese, Frank. Efficient Implementation of Approximate Fourth Order N-Electron Valence State Perturbation Theory. *J. Chem. Theory Comput.*, **2025**, 21 (8), 3953–3967. DOI: 10.1021/acs.jctc.4c01735.
4. Guo, Yang; Sivalingam, Kantharuban; Chilkuri, Vijay Gopal; Neese, Frank. Approximations of density matrices in N-electron valence state second-order perturbation theory (NEVPT2). III. Large active space calculations with selected configuration interaction reference. *J. Chem. Phys.*, **2025**, 162 (14), 144110. DOI: 10.1063/5.0262473.
5. Helmich-Paris, Benjamin; Kjellgren, Erik Rosendahl; Jensen, Hans Jørgen Aa. Excited-State Methods Based on State-Averaged Long-Range CASSCF Short-Range DFT. under review in Phys. Chem. Chem. Phys., **2025**.
6. de Souza, Bernardo. GOAT: A Global Optimization Algorithm for Molecules and Atomic Clusters. *Angew. Chem. Int. Ed.*, **2025**, 64 (18), e202500393. DOI: 10.1002/anie.202500393.

7. Leyser da Costa Gouveia, Tiago; Maganas, Dimitrios; Neese, Frank. General Spin-Restricted Open-Shell Configuration Interaction Approach: Application to Metal K-Edge X-ray Absorption Spectra of Ferro- and Antiferromagnetically Coupled Dimers. *The Journal of Physical Chemistry A*, **2025**, 129 (1), 330–345. PMID: 39680653. arXiv:https://doi.org/10.1021/acs.jpca.4c05228, DOI: 10.1021/acs.jpca.4c05228.

8. Casanova-Páez, M.; Neese, F. Core-Excited States for Open-Shell Systems in Similarity-Transformed Equation-of-Motion Theory. *J. Chem. Theory Comput.*, **2025**, 21 (3), 1306–1321. DOI: 10.1021/acs.jctc.4c01181.

9. Regni, Gianluca; Baldinelli, Lorenzo; Bistoni, Giovanni. A Quantum Chemical Method for Dissecting London Dispersion Energy into Atomic Building Blocks (In press). **2025**. DOI: 10.1021/acscentsci.5c00356.

10. Colinet, Pauline; Neese, Frank; Helmich-Paris, Benjamin. Improving the Efficiency of Electrostatic Embedding Using the Fast Multipole Method. *J. Comput. Chem.*, **2025**, 46 (1), e27532. DOI: 10.1002/jcc.27532.

11. Neese, Frank; Colinet, Pauline; DeSouza, Bernardo; Helmich-Paris, Benjamin; Wennmohs, Frank; Becker, Ute. The "Bubblepole" (BUPO) Method for Linear-Scaling Coulomb Matrix Construction with or without Density Fitting. *J. Phys. Chem. A*, **2025**, 129 (10), 2618–2637. DOI: 10.1021/acs.jpca.4c07415.

12. Lang, Lucas; Chilkuri, Vijay Gopal; Neese, Frank. Treating Spin–Orbit Coupling and Spin–Spin Coupling in the Framework of the Iterative Configuration Expansion Selected CI method (In press). **2025**. DOI: 10.1021/acs.jctc.5c00463.

B.1.2 2024

1. Leyser da Costa Gouveia, Tiago; Maganas, Dimitrios; Neese, Frank. Restricted Open-Shell Hartree–Fock Method for a General Configuration State Function Featuring Arbitrarily Complex Spin-Couplings. *J. Phys. Chem. A*, **2024**, 128 (25), 5041–5053. PMID: 38886177. DOI: 10.1021/acs.jpca.4c00688.

2. Lechner, M. H.; Papadopoulos, A.; Sivalingam, K.; Auer, A. A.; Koslowski, A.; Becker, U.; Wennmohs, F.; Neese, F. Code generation in ORCA: Progress, Efficiency and Tight integration. *Phys. Chem. Chem. Phys.*, **2024**, 26 (21), 15205–15220.

3. Rao, Shashank V.; Maganas, Dimitrios; Sivalingam, Kantharuban; Atanasov, Mihail; Neese, Frank. Extended Active Space Ab Initio Ligand Field Theory: Applications to Transition-Metal Ions. *Inorg. Chem.*, **2024**, 63 (52), 24672–24684. Publisher: American Chemical Society. DOI: 10.1021/acs.inorgchem.4c03893.

4. Casanova-Páez, M.; Neese, F. Assessment of the similarity-transformed equation of motion (STEOM) for open-shell organic and transition metal molecules. *J. Chem. Phys.*, **2024**, 161, 1444120–XXXX. DOI: 10.1063/5.0234225.

5. Wittmann, Lukas; Gordiy, Igor; Friede, Marvin; Helmich-Paris, Benjamin; Grimme, Stefan; Hansen, Andreas; Bursch, Markus. Extension of the D3 and D4 London dispersion corrections to the full actinides series. *Phys. Chem. Chem. Phys.*, **2024**, 26, 21379–21394. DOI: 10.1039/D4CP01514B.

B.1.3 2023

1. Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.

2. Foglia, Nicolás; De Souza, Bernardo; Maganas, Dimitrios; Neese, Frank. Including vibrational effects in magnetic circular dichroism spectrum calculations in the framework of excited state dynamics. *J. Chem. Phys.*, **2023**, 158 (15), 154108. DOI: 10.1063/5.0144845.

3. Neese, Frank. The SHARK integral generation and digestion system. *J. Comp. Chem.*, **2023**, 44 (3), 381–396. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.26942, DOI: 10.1002/jcc.26942.

B.1.4 2022

1. Helmich-Paris, Benjamin. A trust-region augmented Hessian implementation for state-specific and state-averaged CASSCF wave functions. *J. Chem. Phys.*, **2022**, 156 (20), 204104. Publisher: American Institute of Physics. DOI: 10.1063/5.0090447.

2. Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.

3. Foglia, Nicolás O.; Maganas, Dimitrios; Neese, Frank. Going beyond the electric-dipole approximation in the calculation of absorption and (magnetic) circular dichroism spectra including scalar relativistic and spin–orbit coupling effects. *J. Chem. Phys.*, **2022**, 157 (8), 084120. arXiv:10.1063/5.0094709, DOI: 10.1063/5.0094709.

B.1.5 2021

1. Helmich-Paris, Benjamin. A Trust-Region Augmented Hessian Implementation for Restricted and Unrestricted Hartree–Fock and Kohn–Sham Methods. *J. Chem. Phys.*, **2021**, 154 (16), 164104. DOI: 10.1063/5.0040798.

2. Helmich-Paris, Benjamin; de Souza, Bernardo; Neese, Frank; Izsák, Róbert. An improved chain of spheres for exchange algorithm. *J. Chem. Phys.*, **2021**, 155 (10), 104109. DOI: 10.1063/5.0058766.

3. Garcia-Ratés, M.; Becker, U.; Neese, F. Implicit Solvation in Domain Based Pair Natural Orbital Coupled Cluster (DLPNO-CCSD) Theory. *J. Comput. Chem.*, **2021**, 42 (27), 1959–1973. DOI: 10.1002/jcc.26726.

4. Stoychev, Georgi L.; Auer, Alexander A.; Gauss, Jürgen; Neese, Frank. DLPNO-MP2 Second Derivatives for the Computation of Polarizabilities and NMR Shieldings. *J. Chem. Phys.*, **2021**, 154 (16), 164110. DOI: 10.1063/5.0047125.

5. Guo, Y.; Li, W.; Li, S. Improved Cluster-in-Molecule Local Correlation Approach for Electron Correlation Calculation of Large Systems. *J. Phys. Chem. A*, **2014**, 118 (39), 8996–9004. DOI: 10.1021/jp501976x.

6. Guo, Yang; Sivalingam, Kantharuban; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). I. Revisiting the NEVPT2 Construction. *J. Chem. Phys.*, **2021**, 154 (21), 214111. DOI: 10.1063/5.0051211.

7. Guo, Yang; Sivalingam, Kantharuban; Kollmar, Christian; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). II. The Full Rank NEVPT2 (FR-NEVPT2) Formulation. *J. Chem. Phys.*, **2021**, 154 (21), 214113. DOI: 10.1063/5.0051218.

8. Chilkuri, Vijay Gopal; Neese, Frank. Comparison of Many-Particle Representations for Selected-CI I: A Tree Based Approach. *J. Comput. Chem.*, **2021**, 42 (14), 982–1005. DOI: 10.1002/jcc.26518.

9. Chilkuri, Vijay Gopal; Neese, Frank. Comparison of Many-Particle Representations for Selected Configuration Interaction: II. Numerical Benchmark Calculations. *J. Chem. Theory Comput.*, **2021**, 17 (5), 2868–2885. DOI: 10.1021/acs.jctc.1c00081.

10. Kollmar, Christian; Sivalingam, Kantharuban; Guo, Yang; Neese, Frank. An efficient implementation of the NEVPT2 and CASPT2 methods avoiding higher-order density matrices. *J. Chem. Phys.*, **2021**, 155 (23), 234104. DOI: 10.1063/5.0072129.

11. Helmich-Paris, Benjamin. Simulating X-ray absorption spectra with complete active space self-consistent field linear response methods. *Int. J. Quantum Chem.*, **2021**, 121 (3), e26559. DOI: 10.1002/qua.26559.

12. Ni, Z. G.; Guo, Y.; Neese, F.; Li, W.; Li, S. H. Cluster-in-Molecule Local Correlation Method with an Accurate Distant Pair Correction for Large Systems. *J. Chem. Theo. Comp.*, **2021**, 17, 756–766.

B.1.6 2020

1. Rolfes, Julian D.; Neese, Frank; Pantazis, Dimitrios A. All-Electron Scalar Relativistic Basis Sets for the Elements Rb–Xe. *J. Comput. Chem.*, **2020**, 41, 1842–1849. DOI: 10.1002/jcc.26355.

2. Garcia-Ratés, M.; Neese, F. Effect of the Solute Cavity on the Solvation Energy and its Derivatives within the Framework of the Gaussian Charge Scheme. *J. Comput. Chem.*, **2020**, 41 (9), 922–939. DOI: 10.1002/jcc.26139.

3. Tran, Van Anh; Neese, Frank. Double-Hybrid Density Functional Theory for g-Tensor Calculations Using Gauge Including Atomic Orbitals. *J. Chem. Phys.*, **2020**, 153 (5), 054105. DOI: 10.1063/5.0013799.

4. Lang, Lucas; Atanasov, Mihail; Neese, Frank. Improvement of Ab Initio Ligand Field Theory by Means of Multistate Perturbation Theory. *J. Phys. Chem. A*, **2020**, 124 (5), 1025–1037. DOI: 10.1021/acs.jpca.9b11227.

5. Lang, Lucas; Sivalingam, Kantharuban; Neese, Frank. The Combination of Multipartitioning of the Hamiltonian with Canonical Van Vleck Perturbation Theory Leads to a Hermitian Variant of Quasidegenerate N-Electron Valence Perturbation Theory. *J. Chem. Phys.*, **2020**, 152 (1), 014109. DOI: 10.1063/1.5133746.

6. Kollmar, Christian; Sivalingam, Kantharuban; Neese, Frank. An Alternative Choice of the Zeroth-Order Hamiltonian in CASPT2 Theory. *J. Chem. Phys.*, **2020**, 152 (21), 214110. DOI: 10.1063/5.0010019.

7. Lang, Lucas; Ravera, Enrico; Parigi, Giacomo; Luchinat, Claudio; Neese, Frank. Solution of a Puzzle: High-Level Quantum-Chemical Treatment of Pseudocontact Chemical Shifts Confirms Classic Semiempirical Theory. *J. Phys. Chem. Lett.*, **2020**, 11 (20), 8735–8744.

8. Altun, A.; Neese, F.; Bistoni, G. Extrapolation to the Limit of a Complete Pair Natural Orbital Space in Local Coupled-Cluster Calculations. *J. Chem. Theory Comput.*, **2020**, 16 (10), 6142–6149. DOI: 10.1021/acs.jctc.0c00344.

9. Auer, A. A.; Tran, V. A.; Sharma, B.; Stoychev, G. L.; Marx, D.; Neese, F. A case study of density functional theory and domain-based local pair natural orbital coupled cluster for vibrational effects on EPR hyperfine coupling constants: vibrational perturbation theory versus ab initio molecular dynamics. *Mol. Phys.*, **2020**, 118 (19-20), e1797916. DOI: 10.1080/00268976.2020.1797916.

10. Datta, Dipayan; Saitow, Masaaki; Sandhöfer, Barbara; Neese, Frank. 57Fe Mössbauer parameters from domain based local pair-natural orbital coupled-cluster theory. *J. Chem. Phys.*, **2020**, 153 (20), 204101. DOI: 10.1063/5.0022215.

11. Dittmer, A.; Stoychev, G. L.; Maganas, D.; Auer, A. A.; Neese, F. Computation of NMR Shielding Constants for Solids Using an Embedded Cluster Approach with DFT, Double-Hybrid DFT, and MP2. *J. Chem. Theory Comput.*, **2020**, 16 (11), 6950–6967. DOI: 10.1021/acs.jctc.0c00067.

12. Guo, Yang; Riplinger, Christoph; Liakos, Dimitrios G.; Becker, Ute; Saitow, Masaaki; Neese, Frank. Linear scaling perturbative triples correction approximations for open-shell domain-based local pair natural orbital coupled cluster singles and doubles theory [DLPNO-CCSD(T/T)]. *J. Chem. Phys.*, **2020**, 152 (2), 024116. DOI: 10.1063/1.5127550.

13. Kumar, A.; Neese, F.; Valeev, E. F. Explicitly correlated coupled cluster method for accurate treatment of open-shell molecules with hundreds of atoms. *J. Chem. Phys.*, **2020**, 153 (9), 17. DOI: 10.1063/5.0012753.

14. Liakos, D. G.; Guo, Y.; Neese, F. Comprehensive Benchmark Results for the Domain Based Local Pair Natural Orbital Coupled Cluster Method (DLPNO-CCSD(T)) for Closed- and Open-Shell Systems. *J. Phys. Chem. A*, **2020**, 124 (1), 90–100. DOI: 10.1021/acs.jpca.9b05734.

15. Neese, Frank; Wennmohs, Frank; Becker, Ute; Riplinger, Christoph. The ORCA quantum chemistry program package. *J. Chem. Phys.*, **2020**, 152 (22), 224108. DOI: https://aip.scitation.org/doi/10.1063/5.0004608.

B.1.7 2019

1. Pinski, Peter; Neese, Frank. Analytical Gradient for the Domain-Based Local Pair Natural Orbital Second Order Møller-Plesset Perturbation Theory Method (DLPNO-MP2). *J. Chem. Phys.*, **2019**, 150, 164102.
2. Kollmar, Christian; Sivalingam, Kantharuban; Helmich-Paris, Benjamin; Angeli, Celestino; Neese, Frank. A perturbation-based super-CI approach for the orbital optimization of a CASSCF wave function. *J. Comput. Chem.*, **2019**, 40 (14), 1463–1470. DOI: 10.1002/jcc.25801.
3. Lang, Lucas; Neese, Frank. Spin-Dependent Properties in the Framework of the Dynamic Correlation Dressed Complete Active Space Method. *J. Chem. Phys.*, **2019**, 150 (10), 104104. DOI: 10.1063/1.5085203.
4. Maganas, Dimitrios; Kowalska, Joanna K.; Nooijen, Marcel; DeBeer, Serena; Neese, Frank. Comparison of Multireference Ab Initio Wavefunction Methodologies for X-Ray Absorption Edges: A Case Study on $[Fe(II/III)Cl_4]^{2-}/^{1-}$ Molecules. *J. Chem. Phys.*, **2019**, 150 (10), 104106. DOI: 10.1063/1.5051613.
5. Helmich-Paris, Benjamin. CASSCF Linear Response Calculations for Large Open-Shell Molecules. *J. Chem. Phys.*, **2019**, 150 (17), 174121. DOI: 10.1063/1.5092613.
6. de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Efficient Simulation of Overtones and Combination Bands in Resonant Raman Spectra. *J. Chem. Phys.*, **2019**, 150 (21), 044105. DOI: 10.1063/1.5099247.
7. Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. HFLD: A Nonempirical London Dispersion-Corrected Hartree–Fock Method for the Quantification and Analysis of Noncovalent Interaction Energies of Large Molecular Systems. *J. Chem. Theory Comput.*, **2019**, 15 (11), 5894–5907. arXiv:10.1021/acs.jctc.9b00425, DOI: 10.1021/acs.jctc.9b00425.
8. Altun, A.; Neese, F.; Bistoni, G. Open-Shell Variant of the London Dispersion-Corrected Hartree-Fock Method HFLD for the Quantification and Analysis of Noncovalent Interaction Energies. *J. Chem. Theory Comput.*, **2022**, 18 (4), 2292–2307. DOI: 10.1021/acs.jctc.1c01295.
9. Dutta, Achintya Kumar; Saitow, Masaaki; Demoulin, Baptiste; Neese, Frank; Izsák, Róbert. A domain-based local pair natural orbital implementation of the equation of motion coupled cluster method for electron attached states. *J. Chem. Phys.*, **2019**, 150 (16), 164123. DOI: 10.1063/1.5089637.
10. Garcia-Ratés, M.; Neese, F. Efficient implementation of the analytical second derivatives of hartree-fock and hybrid DFT energies within the framework of the conductor-like polarizable continuum model. *J. Comput. Chem.*, **2019**, 40 (20), 1816–1828. DOI: 10.1002/jcc.25833.
11. Haldar, S.; Riplinger, C.; Demoulin, B.; Neese, F.; Izsak, R.; Dutta, A. K. Multilayer Approach to the IP-EOM-DLPNO-CCSD Method: Theory, Implementation, and Application. *J. Chem. Theory Comput.*, **2019**, 15 (4), 2265–2277. DOI: 10.1021/acs.jctc.8b01263.
12. Lang, J.; Brabec, J.; Saitow, M.; Pittner, J.; Neese, F.; Demel, O. Perturbative triples correction to domain-based local pair natural orbital variants of Mukherjee's state specific coupled cluster method. *Phys. Chem. Chem. Phys.*, **2019**, 21 (9), 5022–5038. DOI: 10.1039/c8cp03577f.
13. Saitow, M.; Dutta, A. K.; Neese, F. Accurate Ionization Potentials, Electron Affinities and Electronegativities of Single-Walled Carbon Nanotubes by State-of-the-Art Local Coupled-Cluster Theory. *Bull. Chem. Soc. Jpn.*, **2019**, 92 (1), 170–174. DOI: 10.1246/bcsj.20180254.

B.1.8 2018

1. Stoychev, Georgi L.; Auer, Alexander A.; Neese, Frank. Efficient and Accurate Prediction of Nuclear Magnetic Resonance Shielding Tensors with Double-Hybrid Density Functional Theory. *J. Chem. Theory Comput.*, **2018**, 14 (9), 4756–4771. DOI: 10.1021/acs.jctc.8b00624.
2. Pinski, Peter; Neese, Frank. Communication: Exact Analytical Derivatives for the Domain-Based Local Pair Natural Orbital MP2 Method (DLPNO-MP2). *J. Chem. Phys.*, **2018**, 148, 031101. DOI: 10.1063/1.5011204.
3. Guo, Yang; Becker, Ute; Neese, Frank. Comparison and Combination of "Direct" and Fragment Based Local Correlation Methods: Cluster in Molecules and Domain Based Local Pair Natural Orbital Perturbation and Coupled Cluster Theories. *J. Chem. Phys.*, **2018**, 148 (12), 124117. DOI: 10.1063/1.5021898.

4. Guo, Yang; Riplinger, Christoph; Becker, Ute; Liakos, Dimitrios G.; Minenkov, Yury; Cavallo, Luigi; Neese, Frank. Communication: An Improved Linear Scaling Perturbative Triples Correction for the Domain Based Local Pair-Natural Orbital Based Singles and Doubles Coupled Cluster Method [DLPNO-CCSD(T)]. *J. Chem. Phys.*, **2018**, 148 (1), 011101. DOI: 10.1063/1.5011798.

5. de Souza, Bernardo; Neese, Frank; Izsak, Robert. On the Theoretical Prediction of Fluorescence Rates from First Principles Using the Path Integral Approach. *J. Chem. Phys.*, **2018**, 148 (3), 034104. DOI: 10.1063/1.5010895.

6. Maganas, D.; DeBeer, S.; Neese, F. Pair Natural Orbital Restricted Open-Shell Configuration Interaction (PNO-ROCIS) Approach for Calculating x-Ray Absorption Spectra of Large Chemical Systems. *J. Phys. Chem. A*, **2018**, 122 (5), 1215–1227. DOI: 10.1021/acs.jpca.7b10880.

7. Stoychev, Georgi L; Auer, Alexander A; Izsák, Róbert; Neese, Frank. Self-Consistent Field Calculation of Nuclear Magnetic Resonance Chemical Shielding Constants Using Gauge-Including Atomic Orbitals and Approximate Two-Electron Integrals. *J. Chem. Theory Comput.*, **2018**, 14 (2), 619–637. DOI: 10.1021/acs.jctc.7b01006.

8. Saitow, Masaaki; Neese, Frank. Accurate Spin-Densities Based on the Domain-Based Local Pair-Natural Orbital Coupled-Cluster Theory. *J. Chem. Phys.*, **2018**, 149, 034104. DOI: 10.1063/1.5027114.

9. Bistoni, G.; Polyak, I.; Sparta, M.; Thiel, W.; Neese, F. Toward Accurate QM/MM Reaction Barriers with Large QM Regions Using Domain Based Pair Natural Orbital Coupled Cluster Theory. *J. Chem. Theory Comput.*, **2018**, 14 (7), 3524–3531. DOI: 10.1021/acs.jctc.8b00348.

10. Brabec, J.; Lang, J.; Saitow, M.; Pittner, J.; Neese, F.; Demel, O. Domain-Based Local Pair Natural Orbital Version of Mukherjee's State-Specific Coupled Cluster Method. *J. Chem. Theory Comput.*, **2018**, 14 (3), 1370–1382. DOI: 10.1021/acs.jctc.7b01184.

11. Dutta, A. K.; Neese, F.; Izsak, R. Accelerating the Coupled-Cluster Singles and Doubles Method Using the Chain-of-Sphere Approximation. *Mol. Phys.*, **2018**, 116 (11), 1428–1434. DOI: 10.1080/00268976.2017.1416201.

12. Dutta, A. K.; Saitow, M.; Riplinger, C.; Neese, F.; Izsak, R. A Nearlinear Scaling Equation of Motion Coupled Cluster Method for Ionized States. *J. Chem. Phys.*, **2018**, 148 (24), 13. DOI: 10.1063/1.5029470.

13. Dutta, A. K.; Nooijen, M.; Neese, F.; Izsak, R. Exploring the Accuracy of a Low Scaling Similarity Transformed Equation of Motion Method for Vertical Excitation Energies. *J. Chem. Theory Comput.*, **2018**, 14 (1), 72–91. DOI: 10.1021/acs.jctc.7b00802.

14. Melo, M. C. R.; Bernardi, R. C.; Rudack, T.; Scheurer, M.; Riplinger, C.; Phillips, J. C.; Maia, J. D. C.; Rocha, G. B.; Ribeiro, J. V.; Stone, J. E.; Neese, F.; Schulten, K.; Luthey-Schulten, Z. NAMD Goes Quantum: An Integrative Suite for Hybrid Simulations. *Nat. Methods*, **2018**, 15 (5), 351–354. DOI: 10.1038/nmeth.4638.

15. Neese, Frank. Software update: the ORCA program system, version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2018**, 8 (1), e1327. DOI: http://doi.wiley.com/10.1002/wcms.1327.

16. Sen, Avijit; de Souza, Bernardo; Huntington, Lee M. J.; Krupička, Martin; Neese, Frank; Izsák, Róbert. An Efficient Pair Natural Orbital Based Configuration Interaction Scheme for the Calculation of Open-Shell Ionization Potentials. *J. Chem. Phys.*, **2018**, 149 (11), 114108.

B.1.9 2017

1. Stoychev, Georgi L.; Auer, Alexander A.; Neese, Frank. Automatic Generation of Auxiliary Basis Sets. *J. Chem. Theory Comput.*, **2017**, 13 (2), 554. DOI: 10.1021/acs.jctc.6b01041.

2. Sparta, Manuel; Retegan, Marius; Pinski, Peter; Riplinger, Christoph; Becker, Ute; Neese, Frank. Multilevel Approaches within the Local Pair Natural Orbital Framework. *J. Chem. Theory Comput.*, **2017**, 13 (7), 3198–3207. DOI: 10.1021/acs.jctc.7b00260.

3. Krupička, Martin; Sivalingam, Kantharuban; Huntington, Lee; Auer, Alexander A.; Neese, Frank. A toolchain for the automatic generation of computer codes for correlated wavefunction calculations. *J. Comput. Chem.*, **2017**, 38 (21), 1853–1868. DOI: 10.1002/jcc.24833.

4. Saitow, M.; Becker, U.; Riplinger, C.; Valeev, E. F.; Neese, F. *J. Chem. Phys.*, **2017**, 146, 164105.

5. Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. Explicitly Correlated N-Electron Valence State Perturbation Theory (NEVPT2-F12). *J. Chem. Phys.*, **2017**, 147 (6), 064110. DOI: 10.1063/1.4996560.

6. Pathak, Shubhrodeep; Lang, Lucas; Neese, Frank. A Dynamic Correlation Dressed Complete Active Space Method: Theory, Implementation, and Preliminary Applications. *J. Chem. Phys.*, **2017**, 147, 234109.

7. Huntington, L .M. J.; Krupička, M.; Neese, F.; Izsák, R. Similarity transformed equation of motion coupled-cluster theory based on an unrestricted Hartree-Fock reference for applications to high-spin open-shell systems. *J. Chem. Phys.*, **2017**, 147, 174104.

8. Dutta, Achintya Kumar; Nooijen, Marcel; Neese, Frank; Izsák, Róbert. Automatic active space selection for the similarity transformed equations of motion coupled cluster method. *J. Chem. Phys.*, **2017**, 146 (7), 074103.

9. Dutta, Achintya Kumar; Neese, Frank; Izsák, Róbert. A simple scheme for calculating approximate transition moments within the equation of motion expectation value formalism. *J. Chem. Phys.*, **2017**, 146 (21), 214111.

10. Grimme, Stefan; Bannwarth, Christoph; Dohm, Sebastian; Hansen, Andreas; Pisarek, Jana; Pracht, Philipp; Seibert, Jakob; Neese, Frank. Fully Automated Quantum-Chemistry-Based Computation of Spin–Spin-Coupled Nuclear Magnetic Resonance Spectra. *Angew. Chem. Int. Ed.*, **2017**, 56 (46), 14763–14769. DOI: 10.1002/anie.201708266.

11. Kalinowski, Jaroslaw; Wennmohs, Frank; Neese, Frank. Arbitrary angular momentum electron repulsion integrals with graphical processing units: Application to the resolution of identity hartree–fock method. *J. Chem. Theory Comput.*, **2017**, 13 (7), 3160–3170.

12. Maganas, Dimitrios; DeBeer, Serena; Neese, Frank. A Restricted Open Configuration Interaction with Singles Method To Calculate Valence-to-Core Resonant X-ray Emission Spectra: A Case Study. *Inorg. Chem.*, **2017**, 56 (19), 11819–11836. DOI: 10.1021/acs.inorgchem.7b01810.

13. Mai, Sebastian; Plasser, Felix; Pabst, Mathias; Neese, Frank; Köhn, Andreas; González, Leticia. Surface Hopping Dynamics Including Intersystem Crossing Using the Algebraic Diagrammatic Construction Method. *J. Chem. Phys.*, **2017**, 147 (18), 184109. DOI: 10.1063/1.4999687.

14. Pavošević, Fabijan; Peng, Chong; Pinski, Peter; Riplinger, Christoph; Neese, Frank; Valeev, Edward F. SparseMaps—A systematic infrastructure for reduced scaling electronic structure methods. V. Linear scaling explicitly correlated coupled-cluster method with pair natural orbitals. *J. Chem. Phys.*, **2017**, 146 (17), 174108.

15. Veis, Libor; Antalík, Andrej; Brabec, Jiri; Neese, Frank; Legeza, Ors; Pittner, Jiri. Coupled cluster method with single and double excitations tailored by matrix product state wave functions. *J. Phys. Chem. Lett.*, **2016**, 7 (20), 4072–4078.

B.2 Relevant Applications, Benchmarks and Reviews

B.2.1 2024

1. Gray, Montgomery; Herbert, John M. Assessing the domain-based local pair natural orbital (DLPNO) approximation for non-covalent interactions in sizable supramolecular complexes. *J. Chem. Phys.*, **2024**, 161 (5), 054114. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0206533/20096343/054114_1_5.0206533.pdf, DOI: 10.1063/5.0206533.

B.2.2 2023

1. Neugebauer, Hagen; Pinski, Peter; Grimme, Stefan; Neese, Frank; Bursch, Markus. Assessment of DLPNO-MP2 Approximations in Double-Hybrid DFT. *J. Chem. Theory Comput.*, **2023**, 19 (21), 7695–7703. DOI: 10.1021/acs.jctc.3c00896.
2. Wappett, Dominique A.; Goerigk, Lars. Exploring CPS-Extrapolated DLPNO–CCSD(T1) Reference Values for Benchmarking DFT Methods on Enzymatically Catalyzed Reactions. *J. Phys. Chem. A*, **2024**, 128 (1), 62–72. DOI: 10.1021/acs.jpca.3c05086.
3. Wang, Zikuan; Neese, Frank. Development of NOTCH, an all-electron, beyond-NDDO semiempirical method: Application to diatomic molecules. *The Journal of Chemical Physics*, **2023**, 158 (18), 184102. DOI: 10.1063/5.0141686.

B.2.3 2022

1. Bursch, Markus; Mewes, Jan-Michael; Hansen, Andreas; Grimme, Stefan. Best-Practice DFT Protocols for Basic Molecular Computational Chemistry. *Angew. Chem. Int. Ed.*, **2022**, 61 (42), e202205735. DOI: 10.1002/anie.202205735.

B.2.4 2021

1. Brémond, Éric; Sancho-García, Juan Carlos; Pérez-Jiménez, Ángel José; Adamo, Carlo. *J. Chem. Phys.*, **2014**, 141, 031101.
2. Beck, M. E.; Riplinger, C.; Neese, F.; Bistoni, G. Unraveling Individual Host-Guest Interactions in Molecular Recognition from First Principles Quantum Mechanics: Insights into the Nature of Nicotinic Acetylcholine Receptor Agonist Binding. *J. Comput. Chem.*, **2021**, 42 (5), 293–302. DOI: 10.1002/jcc.26454.
3. Berraud-Pache, R.; Santamaria-Aranda, E.; de Souza, B.; Bistoni, G.; Neese, F.; Sampedro, D.; Izsak, R. Redesigning Donor-Acceptor Stenhouse Adduct Photoswitches through a Joint Experimental and Computational Study. *Chem. Sci.*, **2021**, 12 (8), 2916–2924. DOI: 10.1039/d0sc06575g.
4. Bursch, Markus; Hansen, Andreas; Pracht, Philipp; Kohn, Julia T.; Grimme, Stefan. Theoretical study on conformational energies of transition metal complexes. *Phys. Chem. Chem. Phys.*, **2021**, 23, 287–299. DOI: 10.1039/D0CP04696E.
5. Daniel, C.; Gonzalez, L.; Neese, F. Quantum Theory: The Challenge of Transition Metal Complexes. *Phys. Chem. Chem. Phys.*, **2021**, 23 (4), 2533–2534. DOI: 10.1039/d0cp90278k.
6. Maurer, Leonard R.; Bursch, Markus; Grimme, Stefan; Hansen, Andreas. Assessing Density Functional Theory for Chemically Relevant Open-Shell Transition Metal Reactions. *J. Chem. Theory Comput.*, **2021**, 17 (10), 6134–6151. DOI: 10.1021/acs.jctc.1c00659.
7. Schulz, C. E.; van Gastel, M.; Pantazis, D. A.; Neese, F. Converged Structural and Spectroscopic Properties for Refined QM/MM Models of Azurin. *Inorg. Chem.*, **2021**, 60 (10), 7399–7412. DOI: 10.1021/acs.inorgchem.1c00640.
8. Schulz, C. E.; Castillo, R. G.; Pantazis, D. A.; DeBeer, S.; Neese, F. Structure-Spectroscopy Correlations for Intermediate q of Soluble Methane Monooxygenase: Insights from QM/MM Calculations. *J. Am. Chem. Soc.*, **2021**, 143 (17), 6560–6577. DOI: 10.1021/jacs.1c01180.
9. Sirohiwal, A.; Neese, F.; Pantazis, D. A. How Can We Predict Accurate Electrochromic Shifts for Biochromophores? A Case Study on the Photosynthetic Reaction Center. *J. Chem. Theory Comput.*, **2021**, 17 (3), 1858–1873. DOI: 10.1021/acs.jctc.0c01152.
10. Sirohiwal, A.; Neese, F.; Pantazis, D. A. Chlorophyll excitation energies and structural stability of the CP47 antenna of photosystem II: a case study in the first-principles simulation of light-harvesting complexes. *Chem. Sci.*, **2021**, 12 (12), 4463–4476. DOI: 10.1039/d0sc06616h.

11. Tarrago, M.; Romelt, C.; Nehrkorn, J.; Schnegg, A.; Neese, F.; Bill, E.; Ye, S. F. Experimental and Theoretical Evidence for an Unusual Almost Triply Degenerate Electronic Ground State of Ferrous Tetraphenylporphyrin. *Inorg. Chem.*, **2021**, 60 (7), 4966–4985. DOI: 10.1021/acs.inorgchem.1c00031.

B.2.5 2020

1. Rolfes, Julian D.; Neese, Frank; Pantazis, Dimitrios A. All-Electron Scalar Relativistic Basis Sets for the Elements Rb–Xe. *J. Comput. Chem.*, **2020**, 41, 1842–1849. DOI: 10.1002/jcc.26355.
2. Chilkuri, Vijay Gopal; DeBeer, Serena; Neese, Frank. Ligand Field Theory and Angular Overlap Model Based Analysis of the Electronic Structure of Homovalent Iron–Sulfur Dimers. *Inorg. Chem.*, **2020**, 59 (2), 984–995. DOI: 10.1021/acs.inorgchem.9b00974.
3. Sirohiwal, Abhishek; Berraud-Pache, Romain; Neese, Frank; Izsák, Róbert; Pantazis, Dimitrios A. Accurate Computation of the Absorption Spectrum of Chlorophyll a with Pair Natural Orbital Coupled Cluster Methods. *J. Phys. Chem. B*, **2020**, 124 (40), 8761–8771. DOI: 10.1021/acs.jpcb.0c05761.
4. Chakarawet, K.; Atanasov, M.; Marbey, J.; Bunting, P. C.; Neese, F.; Hill, S.; Long, J. R. Strong Electronic and Magnetic Coupling in M-4 (m = Ni, Cu) Clusters via Direct Orbital Interactions between Low-Coordinate Metal Centers. *J. Am. Chem. Soc.*, **2020**, 142 (45), 19161–19169. DOI: 10.1021/jacs.0c08460.
5. Dittmer, A.; Stoychev, G. L.; Maganas, D.; Auer, A. A.; Neese, F. Computation of NMR Shielding Constants for Solids Using an Embedded Cluster Approach with DFT, Double-Hybrid DFT, and MP2. *J. Chem. Theory Comput.*, **2020**, 16 (11), 6950–6967. DOI: 10.1021/acs.jctc.0c00067.
6. Floser, B. M.; Guo, Y.; Riplinger, C.; Tuczek, F.; Neese, F. Detailed Pair Natural Orbital-Based Coupled Cluster Studies of Spin Crossover Energetics. *J. Chem. Theory Comput.*, **2020**, 16 (4), 2224–2235. DOI: 10.1021/acs.jctc.9b01109.
7. Gottschalk, H. C.; Poblotzki, A.; Fatima, M.; Obenchain, D. A.; Perez, C.; Antony, J.; Auer, A. A.; Baptista, L.; Benoit, D. M.; Bistoni, G.; Bohle, F.; Dahmani, R.; Firaha, D.; Grimme, S.; Hansen, A.; Harding, M. E.; Hochlaf, M.; Holzer, C.; Jansen, G.; Klopper, W.; Kopp, W. A.; Kroger, L. C.; Leonhard, K.; Al-Mogren, M. M.; Mouhib, H.; Neese, F.; Pereira, X. N.; Prakash, M.; Ulusoy, I. S.; Mata, R. A.; Suhm, M. A.; Schnell, M. The First Microsolvation Step for Furans: New Experiments and Benchmarking Strategies. *J. Chem. Phys.*, **2020**, 152 (16), 17. DOI: 10.1063/5.0004465.
8. Neugebauer, Hagen; Bohle, Fabian; Bursch, Markus; Hansen, Andreas; Grimme, Stefan. Benchmark Study of Electrochemical Redox Potentials Calculated with Semiempirical and DFT Methods. *J. Phys. Chem. A*, **2020**, 124 (35), 7166–7176. DOI: 10.1021/acs.jpca.0c05052.
9. Hillenbrand, J.; van Gastel, M.; Bill, E.; Neese, F.; Furstner, A. Isolation of a Homoleptic Non-Oxo Mo(V) Alkoxide Complex: Synthesis, Structure, and Electronic Properties of Penta-Tert-Butoxymolybdenum. *J. Am. Chem. Soc.*, **2020**, 142 (38), 16392–16402. DOI: 10.1021/jacs.0c07073.
10. Jung, J.; Loffler, S. T.; Langmann, J.; Heinemann, F. W.; Bill, E.; Bistoni, G.; Scherer, W.; Atanasov, M.; Meyer, K.; Neese, F. Dispersion Forces Drive the Formation of Uranium-Alkane Adducts. *J. Am. Chem. Soc.*, **2020**, 142 (4), 1864–1870. DOI: 10.1021/jacs.9b10620.
11. Maganas, D.; Kowalska, J. K.; Van Stappen, C.; DeBeer, S.; Neese, F. Mechanism of L-2,L-3-Edge x-Ray Magnetic Circular Dichroism Intensity from Quantum Chemical Calculations and Experiment-A Case Study on V-(IV)/V-(III) Complexes. *J. Chem. Phys.*, **2020**, 152 (11), 15. DOI: 10.1063/1.5129029.
12. Meyer, F.; Neese, F. Impact of Modern Spectroscopy in Inorganic Chemistry. *Inorg. Chem.*, **2020**, 59 (19), 13805–13806. DOI: 10.1021/acs.inorgchem.0c02755.
13. Reza Ghafarian Shirazi, Dimitrios A. Pantazis; Neese, Frank. Performance of density functional theory and orbital-optimised second-order perturbation theory methods for geometries and singlet–triplet state splittings of aryl-carbenes. *Mol. Phys.*, **2020**, 118 (21-22), e1764644. DOI: 10.1080/00268976.2020.1764644.
14. Sirohiwal, A.; Neese, F.; Pantazis, D. A. Protein Matrix Control of Reaction Center Excitation in Photosystem II. *J. Am. Chem. Soc.*, **2020**, 142 (42), 18174–18190. DOI: 10.1021/jacs.0c08526.

15. Spiller, N.; Chilkuri, V. G.; DeBeer, S.; Neese, F. Sulfur vs. Selenium as Bridging Ligand in Di-Iron Complexes: A Theoretical Analysis. *Eur. J. Inorg. Chem.*, **2020**, 2020 (15-16), 1525–1538. DOI: 10.1002/ejic.202000033.

16. Yepes, D.; Neese, F.; List, B.; Bistoni, G. Unveiling the Delicate Balance of Steric and Dispersion Interactions in Organocatalysis Using High-Level Computational Methods. *J. Am. Chem. Soc.*, **2020**, 142 (7), 3613–3625. DOI: 10.1021/jacs.9b13725.

B.2.6 2019

1. Maganas, Dimitrios; Kowalska, Joanna K.; Nooijen, Marcel; DeBeer, Serena; Neese, Frank. Comparison of Multireference Ab Initio Wavefunction Methodologies for X-Ray Absorption Edges: A Case Study on $[Fe(II/III)Cl_4]^{2-/1-}$ Molecules. *J. Chem. Phys.*, **2019**, 150 (10), 104106. DOI: 10.1063/1.5051613.

2. Helmich-Paris, Benjamin. Benchmarks for Electronically Excited States with CASSCF Methods. *J. Chem. Theory Comput.*, **2019**, 15 (7), 4170–4179. DOI: 10.1021/acs.jctc.9b00325.

3. Dittmer, Anneke; Izsák, Róbert; Neese, Frank; Maganas, Dimitrios. Accurate Band Gap Predictions of Semiconductors in the Framework of the Similarity Transformed Equation of Motion Coupled Cluster Theory. *Inorg. Chem.*, **2019**, 58 (14), 9303–9315. DOI: 10.1021/acs.inorgchem.9b00994.

4. Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. Effect of Electron Correlation on Intermolecular Interactions: A Pair Natural Orbitals Coupled Cluster Based Local Energy Decomposition Study. *J. Chem. Theory Comput.*, **2019**, 15 (1), 215–228. DOI: 10.1021/acs.jctc.8b00915.

5. Berraud-Pache, R.; Neese, F.; Bistoni, G.; Izsak, R. Computational Design of Near-Infrared Fluorescent Organic Dyes Using an Accurate New Wave Function Approach. *J. Phys. Chem. Lett.*, **2019**, 10 (17), 4822–4828. DOI: 10.1021/acs.jpclett.9b02240.

6. Chang, H. C.; Lin, Y. H.; Werle, C.; Neese, F.; Lee, W. Z.; Bill, E.; Ye, S. F. Conversion of a Fleeting Open-Shell Iron Nitride into an Iron Nitrosyl. *Angew. Chem. Int. Ed.*, **2019**, 58 (49), 17589–17593. DOI: 10.1002/anie.201908689.

7. Chang, H. C.; Mondal, B.; Fang, H. Y.; Neese, F.; Bill, E.; Ye, S. F. Electron Paramagnetic Resonance Signature of Tetragonal Low Spin Iron(V)-Nitrido and -Oxo Complexes Derived from the Electronic Structure Analysis of Heme and Non-Heme Archetypes. *J. Am. Chem. Soc.*, **2019**, 141 (6), 2421–2434. DOI: 10.1021/jacs.8b11429.

8. Bursch, Markus; Caldeweyher, Eike; Hansen, Andreas; Neugebauer, Hagen; Ehlert, Sebastian; Grimme, Stefan. Understanding and Quantifying London Dispersion Effects in Organometallic Complexes. *Acc. Chem. Res.*, **2019**, 52 (1), 258–266. DOI: 10.1021/acs.accounts.8b00505.

9. Keilwerth, M.; Hohenberger, J.; Heinemann, F. W.; Sutter, J.; Scheurer, A.; Fang, H. Y.; Bill, E.; Neese, F.; Ye, S. F.; Meyer, K. A Series of Iron Nitrosyl Complexes Fe-NO(6-9) and a Fleeting Fe-NO(10) Intermediate En Route to a Metalacyclic Iron Nitrosoalkane. *J. Am. Chem. Soc.*, **2019**, 141 (43), 17217–17235. DOI: 10.1021/jacs.9b08053.

10. Krewald, Vera; Neese, Frank; Pantazis, Dimitrios A. Implications of structural heterogeneity for the electronic structure of the final oxygen-evolving intermediate in photosystem II. *J. Inorg. Biochem.*, **2019**, 199, 110797. DOI: 10.1016/j.jinorgbio.2019.110797.

11. Lu, Q.; Neese, F.; Bistoni, G. London Dispersion Effects in the Coordination and Activation of Alkanes in Sigma-Complexes: A Local Energy Decomposition Study. *Phys. Chem. Chem. Phys.*, **2019**, 21 (22), 11569–11577. DOI: 10.1039/c9cp01309a.

12. Neese, F.; Atanasov, M.; Bistoni, G.; Maganas, D.; Ye, S. F. Chemistry and Quantum Mechanics in 2019: Give Us Insight and Numbers. *J. Am. Chem. Soc.*, **2019**, 141 (7), 2814–2824. DOI: 10.1021/jacs.8b13313.

13. Saitow, M.; Dutta, A. K.; Neese, F. Accurate Ionization Potentials, Electron Affinities and Electronegativities of Single-Walled Carbon Nanotubes by State-of-the-Art Local Coupled-Cluster Theory. *Bull. Chem. Soc. Jpn.*, **2019**, 92 (1), 170–174. DOI: 10.1246/bcsj.20180254.

14. Salla, Cristian A. M.; Teixeira dos Santos, Jéssica; Farias, Giliandro; Bortoluzi, Adailton J.; Curcio, Sergio F.; Cazati, Thiago; Izsák, Róbert; Neese, Frank; de Souza, Bernardo; Bechtold, Ivan H. New Boron(III) Blue

Emitters for All-Solution Processed OLEDs: Molecular Design Assisted by Theoretical Modeling. *Eur. J. Inorg. Chem.*, **2019**, 2019 (17), 2247–2257. DOI: 10.1002/ejic.201900265.

15. Shirazi, R. G.; Neese, F.; Pantazis, D. A.; Bistoni, G. Physical Nature of Differential Spin-State Stabilization of Carbenes by Hydrogen and Halogen Bonding: A Domain-Based Pair Natural Orbital Coupled Cluster Study. *J. Phys. Chem. A*, **2019**, 123 (24), 5081–5090. DOI: 10.1021/acs.jpca.9b01051.

16. Sirohiwal, A.; Neese, F.; Pantazis, D. A. Microsolvation of the Redox-Active Tyrosine-d in Photosystem II: Correlation of Energetics with EPR Spectroscopy and Oxidation-Induced Proton Transfer. *J. Am. Chem. Soc.*, **2019**, 141 (7), 3217–3231. DOI: 10.1021/jacs.8b13123.

17. Stavretis, Shelby E.; Cheng, Yongqiang; Daemen, Luke L.; Brown, Craig M.; Moseley, Duncan H.; Bill, Eckhard; Atanasov, Mihail; Ramirez-Cuesta, Anibal J.; Neese, Frank; Xue, Zi-Ling. Probing Magnetic Excitations in CoII Single-Molecule Magnets by Inelastic Neutron Scattering. *Eur. J. Inorg. Chem.*, **2019**, 2019 (8), 1119–1127. DOI: 10.1002/ejic.201801088.

18. Thomsen, M. K.; Nyvang, A.; Walsh, J. P. S.; Bunting, P. C.; Long, J. R.; Neese, F.; Atanasov, M.; Genoni, A.; Oyergaard, J. Insights into Single-Molecule-Magnet Behavior from the Experimental Electron Density of Linear Two-Coordinate Iron Complexes. *Inorg. Chem.*, **2019**, 58 (5), 3211–3218. DOI: 10.1021/acs.inorgchem.8b03301.

B.2.7 2018

1. Altun, A.; Neese, F.; Bistoni, G. Local Energy Decomposition Analysis of Hydrogen-Bonded Dimers within a Domain-Based Pair Natural Orbital Coupled Cluster Study. *Beilstein J. Org. Chem.*, **2018**, 14, 919–929. DOI: 10.3762/bjoc.14.79.

2. Bistoni, G.; Polyak, I.; Sparta, M.; Thiel, W.; Neese, F. Toward Accurate QM/MM Reaction Barriers with Large QM Regions Using Domain Based Pair Natural Orbital Coupled Cluster Theory. *J. Chem. Theory Comput.*, **2018**, 14 (7), 3524–3531. DOI: 10.1021/acs.jctc.8b00348.

3. Bunting, P. C.; Atanasov, M.; Damgaard-Moller, E.; Perfetti, M.; Crassee, I.; Orlita, M.; Overgaard, J.; van Slageren, J.; Neese, F.; Long, J. R. A Linear Cobalt(II) Complex with Maximal Orbital Angular Momentum from a Non-Aufbau Ground State. *Science*, **2018**, 362 (6421), A Linear Cobalt(II) Complex with Ma. DOI: 10.1126/science.aat7319.

4. Chantzis, A.; Kowalska, J. K.; Maganas, D.; DeBeer, S.; Neese, F. Ab Initio Wave Function-Based Determination of Element Specific Shifts for the Efficient Calculation of x-Ray Absorption Spectra of Main Group Elements and First Row Transition Metals. *J. Chem. Theory Comput.*, **2018**, 14 (7), 3686–3702. DOI: 10.1021/acs.jctc.8b00249.

5. Collins, L. R.; van Gastel, M.; Neese, F.; Furstner, A. Enhanced Electrophilicity of Heterobimetallic Bi-Rh Paddlewheel Carbene Complexes: A Combined Experimental, Spectroscopic, and Computational Study. *J. Am. Chem. Soc.*, **2018**, 140 (40), 13042–13055. DOI: 10.1021/jacs.8b08384.

6. David, G.; Wennmohs, F.; Neese, F.; Ferre, N. Chemical Tuning of Magnetic Exchange Couplings Using Broken-Symmetry Density Functional Theory. *Inorg. Chem.*, **2018**, 57 (20), 12769–12776. DOI: 10.1021/acs.inorgchem.8b01970.

7. Gatzenmeier, T.; Turberg, M.; Yepes, D.; Xie, Y. W.; Neese, F.; Bistoni, G.; List, B. Scalable and Highly Diastereo- and Enantioselective Catalytic Diels-Alder Reaction of Alpha,Beta-Unsaturated Methyl Esters. *J. Am. Chem. Soc.*, **2018**, 140 (40), 12671–12676. DOI: 10.1021/jacs.8b07092.

8. Gottschalk, H. C.; Poblotzki, A.; Suhm, M. A.; Al-Mogren, M. M.; Antony, J.; Auer, A. A.; Baptista, L.; Benoit, D. M.; Bistoni, G.; Bohle, F.; Dahmani, R.; Firaha, D.; Grimme, S.; Hansen, A.; Harding, M. E.; Hochlaf, M.; Holzer, C.; Jansen, G.; Klopper, W.; Kopp, W. A.; Kroger, L. C.; Leonhard, K.; Mouhib, H.; Neese, F.; Pereira, M. N.; Ulusoy, I. S.; Wuttke, A.; Mata, R. A. The Furan Microsolvation Blind Challenge for Quantum Chemical Methods: First Steps. *J. Chem. Phys.*, **2018**, 148 (1), 13. DOI: 10.1063/1.5009011.

9. Kubas, Adam; Verkamp, Max; Vura-Weis, Josh; Neese, Frank; Maganas, Dimitrios. Restricted open-shell configuration interaction singles study on M- and L-edge x-ray absorption spectroscopy of solid chemical systems. *J. Chem. Theory Comput.*, **2018**, 14 (8), 4320–4334. DOI: 10.1021/acs.jctc.8b00302.

10. Lu, Q.; Neese, F.; Bistoni, G. Formation of Agostic Structures Driven by London Dispersion. *Angew. Chem. Int. Ed.*, **2018**, 57 (17), 4760–4764. DOI: 10.1002/anie.201801531.

11. Mondal, B.; Neese, F.; Bill, E.; Ye, S. F. Electronic Structure Contributions of Non-Herne Oxo-Iron(v) Complexes to the Reactivity. *J. Am. Chem. Soc.*, **2018**, 140 (30), 9531–9544. DOI: 10.1021/jacs.8b04275.

12. Moseley, D. H.; Stavretis, S. E.; Thirunavukkuarasu, K.; Ozerov, M.; Cheng, Y. Q.; Daemen, L. L.; Ludwig, J.; Lu, Z. G.; Smirnov, D.; Brown, C. M.; Pandey, A.; Ramirez-Cuesta, A. J.; Lamb, A. C.; Atanasov, M.; Bill, E.; Neese, F.; Xue, Z. L. Spin-Phonon Couplings in Transition Metal Complexes with Slow Magnetic Relaxation. *Nature Comm.*, **2018**, 9, 11. DOI: 10.1038/s41467-018-04896-0.

13. Romelt, C.; Ye, S. F.; Bill, E.; Weyhermuller, T.; van Gastel, M.; Neese, F. Electronic Structure and Spin Multiplicity of Iron Tetraphenylporphyrins in Their Reduced States as Determined by a Combination of Resonance Raman Spectroscopy and Quantum Chemistry. *Inorg. Chem.*, **2018**, 57 (4), 2141–2148. DOI: 10.1021/acs.inorgchem.7b03018.

14. Scheibe, B.; Pietzonka, C.; Mustonen, O.; Karppinen, M.; Karttunen, A. J.; Atanasov, M.; Neese, F.; Conrad, M.; Kraus, F. The U2F12 (2-) Anion of Sr U2F12. *Angew. Chem. Int. Ed.*, **2018**, 57 (11), 2914–2918. DOI: 10.1002/anie.201800743.

15. Shirazi, R. G.; Neese, F.; Pantazis, D. A. Accurate Spin-State Energetics for Aryl Carbenes. *J. Chem. Theory Comput.*, **2018**, 14 (9), 4733–4746. DOI: 10.1021/acs.jctc.8b00587.

16. Singh, Saurabh Kumar; Atanasov, Mihail; Neese, Frank. Challenges in multireference perturbation theory for the calculations of the g-tensor of first-row transition-metal complexes. *J. Chem. Theory Comput.*, **2018**, 14 (9), 4662–4677.

17. Van Stappen, C.; Maganas, D.; DeBeer, S.; Bill, E.; Neese, F. Investigations of the Magnetic and Spectroscopic Properties of V(III) and V(IV) Complexes. *Inorg. Chem.*, **2018**, 57 (11), 6421–6438. DOI: 10.1021/acs.inorgchem.8b00486.

18. Yamamoto, K.; Li, J. K.; Garber, J. A. O.; Rolfes, J. D.; Boursalian, G. B.; Borghs, J. C.; Genicot, C.; Jacq, J.; van Gastel, M.; Neese, F.; Ritter, T. Palladium-Catalysed Electrophilic Aromatic C-H Fluorination. *Nature*, **2018**, 554 (7693), 511–514. DOI: 10.1038/nature25749.

B.2.8 2017

1. Jung, Julie; Atanasov, Mihail; Neese, Frank. Ab Initio Ligand-Field Theory Analysis and Covalency Trends in Actinide and Lanthanide Free Ions and Octahedral Complexes. *Inorg. Chem.*, **2017**, 56 (15), 8802–8816. DOI: 10.1021/acs.inorgchem.7b00642.

2. Singh, Saurabh Kumar; Eng, Julien; Atanasov, Mihail; Neese, Frank. Covalency and Chemical Bonding in Transition Metal Complexes: An Ab Initio Based Ligand Field Perspective. *Coordin. Chem. Rev.*, **2017**, 344, 2–25. DOI: 10.1016/j.ccr.2017.03.018.

3. Chakraborty, Uttam; Demeshko, Serhiy; Meyer, Franc; Rebreyend, Christophe; de Bruin, Bas; Atanasov, Mihail; Neese, Frank; Mühldorf, Bernd; Wolf, Robert. Electronic Structure and Magnetic Anisotropy of an Unsaturated Cyclopentadienyl Iron(I) Complex with 15 Valence Electrons. *Angew. Chem. Int. Ed.*, **2017**, 56 (27), 7995–7999. DOI: 10.1002/anie.201702454.

4. Chilkuri, Vijay Gopal; DeBeer, Serena; Neese, Frank. Revisiting the Electronic Structure of FeS Monomers Using Ab Initio Ligand Field Theory and the Angular Overlap Model. *Inorg. Chem.*, **2017**, 56 (17), 10418–10436. DOI: 10.1021/acs.inorgchem.7b01371.

5. Bistoni, Giovanni; Auer, Alexander A.; Neese, Frank. Understanding the Role of Dispersion in Frustrated Lewis Pairs and Classical Lewis Adducts: A Domain-Based Local Pair Natural Orbital Coupled Cluster Study. *Chem. Eur. J.*, **2017**, 23 (4), 865–873.

6. Bistoni, Giovanni; Riplinger, Christoph; Minenkov, Yury; Cavallo, Luigi; Auer, Alexander A.; Neese, Frank. Treating Subvalence Correlation Effects in Domain Based Pair Natural Orbital Coupled Cluster Calculations: An Out-of-the-Box Approach. *J. Chem. Theory Comput.*, **2017**, 13 (7), 3220–3227. DOI: 10.1021/acs.jctc.7b00352.

7. Caldararu, Octav; Olsson, Martin A; Riplinger, Christoph; Neese, Frank; Ryde, Ulf. Binding free energies in the SAMPL5 octa-acid host–guest challenge calculated with DFT-D3 and CCSD (T). *J. Comput.-Aided Mol. Des.*, **2017**, 31 (1), 87–106. DOI: 10.1007/s10822-016-9957-5.

8. Kubas, Adam; Noak, Johannes; Trunschke, Annette; Schlögl, Robert; Neese, Frank; Maganas, Dimitrios. A combined experimental and theoretical spectroscopic protocol for determination of the structure of heterogeneous catalysts: developing the information content of the resonance Raman spectra of M1 MoVOx. *Chem. Sci.*, **2017**, 8 (9), 6338–6353. DOI: 10.1039/C7SC01771E.

9. Minenkov, Yury; Bistoni, Giovanni; Riplinger, Christoph; Auer, Alexander A; Neese, Frank; Cavallo, Luigi. Pair natural orbital and canonical coupled cluster reaction enthalpies involving light to heavy alkali and alkaline earth metals: the importance of sub-valence correlation. *Phys. Chem. Chem. Phys.*, **2017**, 19 (14), 9374–9391.

10. Neese, Frank. High-Level Spectroscopy, Quantum Chemistry, and Catalysis: Not just a Passing Fad. *Angew. Chem. Int. Ed.*, **2017**, 56 (37), 11003–11010.

11. Neese, Frank. *Quantum Chemistry and EPR Parameters*, chapter, pages 1–22. John Wiley & Sons, Ltd, **2017**. DOI: 10.1002/9780470034590.emrstm1505.

12. Pedersen, Kasper S; Woodruff, Daniel N; Singh, Saurabh Kumar; Tressaud, Alain; Durand, Etienne; Atanasov, Mihail; Perlepe, Panagiota; Ollefs, Katharina; Wilhelm, Fabrice; Mathonière, Corine; others. [OsF6] x-: Molecular Models for Spin-Orbit Entangled Phenomena. *Chem. Eur. J.*, **2017**, 23 (47), 11244–11248.

13. Suturina, Elizaveta A; Nehrkorn, Joscha; Zadrozny, Joseph M; Liu, Junjie; Atanasov, Mihail; Weyhermüller, Thomas; Maganas, Dimitrios; Hill, Stephen; Schnegg, Alexander; Bill, Eckhard; others. Magneto-structural correlations in pseudotetrahedral forms of the [Co (SPh) 4] 2–complex probed by magnetometry, MCD spectroscopy, advanced EPR techniques, and ab initio electronic structure calculations. *Inorg. Chem.*, **2017**, 56 (5), 3102–3118.

B.3 Classification by Topic

B.3.1 *Ab initio* Ligand Field Analyis

1. Atanasov, Mihail; Ganyushin, Dmitry; Sivalingam, Kantharuban; Neese, Frank. A Modern First-Principles View on Ligand Field Theory Through the Eyes of Correlated Multireference Wavefunctions. In Mingos, David Michael P.; Day, Peter; Dahl, Jens Peder, editors, *Molecular Electronic Structures of Transition Metal Complexes II*, number 143 in Structure and Bonding, pages 149–220. Springer Berlin Heidelberg, **2011**. DOI: 10.1007/430_2011_57.

2. Lang, Lucas; Atanasov, Mihail; Neese, Frank. Improvement of Ab Initio Ligand Field Theory by Means of Multistate Perturbation Theory. *J. Phys. Chem. A*, **2020**, 124 (5), 1025–1037. DOI: 10.1021/acs.jpca.9b11227.

3. Rao, Shashank V.; Maganas, Dimitrios; Sivalingam, Kantharuban; Atanasov, Mihail; Neese, Frank. Extended Active Space Ab Initio Ligand Field Theory: Applications to Transition-Metal Ions. *Inorg. Chem.*, **2024**, 63 (52), 24672–24684. Publisher: American Chemical Society. DOI: 10.1021/acs.inorgchem.4c03893.

B.3.2 Absorption, Resonance Raman and Fluorescence Spectra

1. Sirohiwal, Abhishek; Berraud-Pache, Romain; Neese, Frank; Izsák, Róbert; Pantazis, Dimitrios A. Accurate Computation of the Absorption Spectrum of Chlorophyll a with Pair Natural Orbital Coupled Cluster Methods. *J. Phys. Chem. B*, **2020**, 124 (40), 8761–8771. DOI: 10.1021/acs.jpcb.0c05761.

2. Petrenko, Taras; Neese, Frank. Analysis and prediction of absorption band shapes, fluorescence band shapes, resonance Raman intensities, and excitation profiles using the time-dependent theory of electronic spectroscopy. *J. Chem. Phys.*, **2007**, 127 (16), 164319. DOI: 10.1063/1.2770706.

3. Petrenko, T.; Neese, F. Efficient and automatic calculation of optical band shapes and resonance Raman spectra for larger molecules within the independent mode displaced harmonic oscillator model. *J. Chem. Phys.*, **2012**, 137, 234107.

4. Petrenko, T.; Krylova, O.; Neese, F.; Sokolowski, M. Optical Absorption and Emission Properties of Rubrene: Insight by a Combined Experimental and Theoretical Study. *New J. Phys.*, **2009**, 11, 015001.

B.3.3 Analytic Hessian Implementation

1. Garcia-Ratés, M.; Neese, F. Efficient implementation of the analytical second derivatives of hartree-fock and hybrid DFT energies within the framework of the conductor-like polarizable continuum model. *J. Comput. Chem.*, **2019**, 40 (20), 1816–1828. DOI: 10.1002/jcc.25833.
2. Bykov, D.; Petrenko, T.; Izsák, R.; Kossmann, S.; Becker, U.; Valeev, E.; Neese, F. Efficient implementation of the analytic second derivatives of Hartree-Fock and hybrid DFT energies: a detailed analysis of different approximations. *Mol. Phys.*, **2015**, 113, 1961.

B.3.4 ANO basis sets

1. Neese, F.; Valeev, E. F. Revisiting the Atomic Natural Orbital Approach for Basis Sets: Robust Systematic Basis Sets for Explicitly Correlated and Conventional Correlated ab initio Methods. *J. Chem. Theory Comput.*, **2011**, 7, 33–43.

B.3.5 Applications

1. Mondal, B.; Neese, F.; Ye, S. F. Toward Rational Design of 3d Transition Metal Catalysts for CO2 Hydrogenation Based on Insights into Hydricity-Controlled Rate-Determining Steps. *Inorg. Chem.*, **2016**, 55, 5438–5444.
2. Mondal, B.; Neese, F.; Ye, S. F. Control in the Rate-Determining Step Provides a Promising Strategy To Develop New Catalysts for CO2 Hydrogenation: A Local Pair Natural Orbital Coupled Cluster Theory Study. *Inorg. Chem.*, **2015**, 54, 7192–7198.
3. Krewald, V.; Retegan, M.; Cox, N.; Messinger, J.; Lubitz, W.; DeBeer, S.; Neese, F.; Pantazis, D. A. Metal oxidation states in biological water splitting. *Chem. Sci.*, **2015**, 6, 1676–1695.
4. Kochem, A.; Weyhermuller, T.; Neese, F.; van Gastel, M. EPR and Quantum Chemical Investigation of a Bioinspired Hydrogenase Model with a Redox-Active Ligand in the First Coordination Sphere. *Organometallics*, **2015**, 34, 995–1000.
5. Kochem, A.; Bill, E.; Neese, F.; van Gastel, M. Mossbauer and computational investigation of a functional NiFe hydrogenase model complex. *Chem. Comm.*, **2015**, 51, 2099–2102.

B.3.6 Approximate FCI

1. Chilkuri, V. G.; Neese, F. Comparison of many-particle representations for selected-CI I: A tree based approach. *J. Comput. Chem.*, **2021**, 42, 982–1005.
2. Chilkuri, V. G.; Neese, F. Comparison of Many-Particle Representations for Selected Configuration Interaction: II. Numerical Benchmark Calculations. *J. Chem. Theory Comput.*, **2021**, 17, 2868–2885.
3. Lang, Lucas; Chilkuri, Vijay Gopal; Neese, Frank. Treating Spin–Orbit Coupling and Spin–Spin Coupling in the Framework of the Iterative Configuration Expansion Selected CI method (In press). **2025**. DOI: 10.1021/acs.jctc.5c00463.

B.3.7 CASSCF/NEVPT2/MRCI & Magnetism

1. Krupička, Martin; Sivalingam, Kantharuban; Huntington, Lee; Auer, Alexander A.; Neese, Frank. A toolchain for the automatic generation of computer codes for correlated wavefunction calculations. *J. Comput. Chem.*, **2017**, 38 (21), 1853–1868. DOI: 10.1002/jcc.24833.
2. Lechner, M. H.; Papadopoulos, A.; Sivalingam, K.; Auer, A. A.; Koslowski, A.; Becker, U.; Wennmohs, F.; Neese, F. Code generation in ORCA: Progress, Efficiency and Tight integration. *Phys. Chem. Chem. Phys.*, **2024**, 26 (21), 15205–15220.
3. Guo, Yang; Sivalingam, Kantharuban; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). I. Revisiting the NEVPT2 Construction. *J. Chem. Phys.*, **2021**, 154 (21), 214111. DOI: 10.1063/5.0051211.
4. Guo, Yang; Sivalingam, Kantharuban; Kollmar, Christian; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). II. The Full Rank NEVPT2 (FR-NEVPT2) Formulation. *J. Chem. Phys.*, **2021**, 154 (21), 214113. DOI: 10.1063/5.0051218.
5. Kempfer, Emily M.; Sivalingam, Kantharuban; Neese, Frank. Efficient Implementation of Approximate Fourth Order N-Electron Valence State Perturbation Theory. *J. Chem. Theory Comput.*, **2025**, 21 (8), 3953–3967. DOI: 10.1021/acs.jctc.4c01735.
6. Kollmar, Christian; Sivalingam, Kantharuban; Helmich-Paris, Benjamin; Angeli, Celestino; Neese, Frank. A perturbation-based super-CI approach for the orbital optimization of a CASSCF wave function. *J. Comput. Chem.*, **2019**, 40 (14), 1463–1470. DOI: 10.1002/jcc.25801.
7. Guo, Yang; Sivalingam, Kantharuban; Chilkuri, Vijay Gopal; Neese, Frank. Approximations of density matrices in N-electron valence state second-order perturbation theory (NEVPT2). III. Large active space calculations with selected configuration interaction reference. *J. Chem. Phys.*, **2025**, 162 (14), 144110. DOI: 10.1063/5.0262473.
8. Lang, Lucas; Sivalingam, Kantharuban; Neese, Frank. The Combination of Multipartitioning of the Hamiltonian with Canonical Van Vleck Perturbation Theory Leads to a Hermitian Variant of Quasidegenerate N-Electron Valence Perturbation Theory. *J. Chem. Phys.*, **2020**, 152 (1), 014109. DOI: 10.1063/1.5133746.
9. Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. Explicitly Correlated N-Electron Valence State Perturbation Theory (NEVPT2-F12). *J. Chem. Phys.*, **2017**, 147 (6), 064110. DOI: 10.1063/1.4996560.
10. Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.
11. Kollmar, Christian; Sivalingam, Kantharuban; Guo, Yang; Neese, Frank. An efficient implementation of the NEVPT2 and CASPT2 methods avoiding higher-order density matrices. *J. Chem. Phys.*, **2021**, 155 (23), 234104. DOI: 10.1063/5.0072129.
12. Kollmar, Christian; Sivalingam, Kantharuban; Neese, Frank. An Alternative Choice of the Zeroth-Order Hamiltonian in CASPT2 Theory. *J. Chem. Phys.*, **2020**, 152 (21), 214110. DOI: 10.1063/5.0010019.
13. Pathak, Shubhrodeep; Lang, Lucas; Neese, Frank. A Dynamic Correlation Dressed Complete Active Space Method: Theory, Implementation, and Preliminary Applications. *J. Chem. Phys.*, **2017**, 147, 234109.
14. Chilkuri, V. G.; DeBeer, S.; Neese, F. Revisiting the Electronic Structure of FeS Monomers Using ab Initio Ligand Field Theory and the Angular Overlap Model. *Inorg. Chem.*, **2017**, 56, 10418.
15. Singh, S. K.; Eng, J.; Atanasov, M.; Neese, F. Covalency and chemical bonding in transition metal complexes: An ab initio based ligand field perspective. *Coord. Chem. Rev.*, **2017**, 344, 225.
16. Jung, J.; Atanasov, M.; Neese, F. Ab Initio Ligand-Field Theory Analysis and Covalency Trends in Actinide and Lanthanide Free Ions and Octahedral Complexes. *Inorg. Chem.*, **2017**, 56, 8802.
17. Werncke, C. G.; Suturina, E.; Bunting, P. C.; Vendier, L.; Long, J. R.; Atanasov, M.; Neese, F.; Sabo-Etienne, S.; Bontemps, S. Homoleptic Two-Coordinate Silylamido Complexes of Chromium(I), Manganese(I), and Cobalt(I). *Chem. Eur. J.*, **2016**, 22, 1668–1674.

18. Rechkemmer, Y.; Breitgoff, F. D.; van der Meer, M.; Atanasov, M.; Hakl, M.; Orlita, M.; Neugebauer, P.; Neese, F.; Sarkar, B.; van Slageren, J. A four-coordinate cobalt(II) single-ion magnet with coercivity and a very high energy barrier. *Nat. Commun.*, **2016**, 7, 10467.

19. Aravena, D.; Atanasov, M.; Neese, F. Periodic Trends in Lanthanide Compounds through the Eyes of Multireference ab Initio Theory. *Inorg. Chem.*, **2016**, 55, 4457–4469.

20. Suturina, E. A.; Maganas, D.; Bill, E.; Atanasov, M.; Neese, F. Magneto-Structural Correlations in a Series of Pseudotetrahedral Co-II(XR)(4) (2-) Single Molecule Magnets: An ab Initio Ligand Field Study. *Inorg. Chem.*, **2015**, 54, 9948–9961.

21. Schweinfurth, D.; Sommer, M. G.; Atanasov, M.; Demeshko, S.; Hohloch, S.; Meyer, F.; Neese, F.; Sarkar, B. The Ligand Field of the Azido Ligand: Insights into Bonding Parameters and Magnetic Anisotropy in a Co(II)-Azido Complex. *J. Am. Chem. Soc.*, **2015**, 137, 1993–2005.

22. Lang, Lucas; Chilkuri, Vijay Gopal; Neese, Frank. Treating Spin–Orbit Coupling and Spin–Spin Coupling in the Framework of the Iterative Configuration Expansion Selected CI method (In press). **2025**. DOI: 10.1021/acs.jctc.5c00463.

B.3.8 Corresponding Orbital Transformation

1. Neese, F. Definition of Corresponding Orbitals and the Diradical Character in Broken Symmetry DFT Calculations on Spin Coupled Systems. *J. Phys. Chem. Solids*, **2004**, 65, 781–785.

B.3.9 COSMO Implementation

1. Sinnecker, S.; Rajendran, A.; Klamt, A.; Diedenhofen, M.; Neese, F. Calculation of Solvent Shifts on Electronic G-Tensors with the Conductor-Like Screening Model (COSMO) and its Self-Consistent Generalization to Real Solvents (COSMO-RS). *J. Phys. Chem. A*, **2006**, 110, 2235–2245.

B.3.10 COSX

1. Dutta, A. K.; Neese, F.; Izsak, R. Speeding up equation of motion coupled cluster theory with the chain of spheres approximation. *J. Chem. Phys.*, **2016**, 144, 034102.

2. Christian, G. J.; Neese, F.; Ye, S. F. Unravelling the Molecular Origin of the Regiospecificity in Extradiol Catechol Dioxygenases. *Inorg. Chem.*, **2016**, 55, 3853–3864.

B.3.11 Coupled-Cluster and Coupled Pair Implementation (MDCI module)

1. Pavošević, F.; Neese, F.; Valeev, E. F. Geminal-spanning orbitals make explicitly correlated reduced-scaling coupled-cluster methods robust, yet simple. *J. Chem. Phys.*, **2014**, 141, 054106.

2. Kollmar, C.; Neese, F. The coupled electron pair approximation: variational formulation and spin adaptation. *Mol. Phys.*, **2010**, 108, 2449–2458.

3. Neese, F.; Wennmohs, F.; Hansen, A.; Grimme, S. Accurate Theoretical Chemistry with Coupled Electron Pair Models. *Acc. Chem. Res.*, **2009**, 42 (5), 641–648.

4. Wennmohs, F.; Neese, F. A Comparative Study of Single Reference Correlation Methods of the Coupled-Pair Type. *Chem. Phys.*, **2008**, 343, 217–230.

B.3.12 EPR/NMR

1. Tran, Van Anh; Neese, Frank. Double-Hybrid Density Functional Theory for g-Tensor Calculations Using Gauge Including Atomic Orbitals. *J. Chem. Phys.*, **2020**, 153 (5), 054105. DOI: 10.1063/5.0013799.
2. Lang, Lucas; Ravera, Enrico; Parigi, Giacomo; Luchinat, Claudio; Neese, Frank. Solution of a Puzzle: High-Level Quantum-Chemical Treatment of Pseudocontact Chemical Shifts Confirms Classic Semiempirical Theory. *J. Phys. Chem. Lett.*, **2020**, 11 (20), 8735–8744.
3. Auer, A. A.; Tran, V. A.; Sharma, B.; Stoychev, G. L.; Marx, D.; Neese, F. A case study of density functional theory and domain-based local pair natural orbital coupled cluster for vibrational effects on EPR hyperfine coupling constants: vibrational perturbation theory versus ab initio molecular dynamics. *Mol. Phys.*, **2020**, 118 (19-20), e1797916. DOI: 10.1080/00268976.2020.1797916.
4. Dittmer, A.; Stoychev, G. L.; Maganas, D.; Auer, A. A.; Neese, F. Computation of NMR Shielding Constants for Solids Using an Embedded Cluster Approach with DFT, Double-Hybrid DFT, and MP2. *J. Chem. Theory Comput.*, **2020**, 16 (11), 6950–6967. DOI: 10.1021/acs.jctc.0c00067.
5. Schulz, C. E.; van Gastel, M.; Pantazis, D. A.; Neese, F. Converged Structural and Spectroscopic Properties for Refined QM/MM Models of Azurin. *Inorg. Chem.*, **2021**, 60 (10), 7399–7412. DOI: 10.1021/acs.inorgchem.1c00640.
6. Sirohiwal, A.; Neese, F.; Pantazis, D. A. Microsolvation of the Redox-Active Tyrosine-d in Photosystem II: Correlation of Energetics with EPR Spectroscopy and Oxidation-Induced Proton Transfer. *J. Am. Chem. Soc.*, **2019**, 141 (7), 3217–3231. DOI: 10.1021/jacs.8b13123.
7. Stoychev, G. L.; Auer, A. A.; Gauss, J.; Neese, F. DLPNO-MP2 second derivatives for the computation of polarizabilities and NMR shieldings. *J. Chem. Phys.*, **2021**, 154, 164110.

B.3.13 ESD Module

Fluorescence

1. de Souza, Bernardo; Neese, Frank; Izsak, Robert. On the Theoretical Prediction of Fluorescence Rates from First Principles Using the Path Integral Approach. *J. Chem. Phys.*, **2018**, 148 (3), 034104. DOI: 10.1063/1.5010895.

Phosphorescence

1. de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Predicting Phosphorescence Rates of Light Organic Molecules Using Time-Dependent Density Functional Theory and the Path Integral Approach to Dynamics. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1896–1904. DOI: 10.1021/acs.jctc.8b00841.

Resonance Raman

1. de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Efficient Simulation of Overtones and Combination Bands in Resonant Raman Spectra. *J. Chem. Phys.*, **2019**, 150 (21), 044105. DOI: 10.1063/1.5099247.

B.3.14 DFT/Hartree–Fock Theory of EPR Parameters

1. Sandhoefer, B.; Neese, F. One-electron contributions to the g-tensor for second-order Douglas–Kroll–Hess theory. *J. Chem. Phys.*, **2012**, 137, 094102.
2. Neese, F. Importance of Direct Spin-Spin Coupling and Spin-Flip Excitations for the Zero-Field Splittings of Transition Metal Complexes: A Case Study. *J. Am. Chem. Soc.*, **2006**, 128, 10213.
3. Neese, F. *J. Chem. Phys.*, **2007**, 127, 164112.
4. Sinnecker, S.; Neese, F. Spin-Spin Contributions to the Zero-Field Splitting Tensor in Organic Triplets, Carbenes and Biradicals – A Density Functional and \it ab initio Study. *J. Phys. Chem. A*, **2006**, 110, 12267.
5. Neese, F. Efficient and Accurate Approximations to the Molecular Spin-Orbit Coupling Operator and their use in Molecular g-Tensor Calculations. *J. Chem. Phys.*, **2005**, 122, 034107.
6. Sinnecker, S.; Rajendran, A.; Klamt, A.; Diedenhofen, M.; Neese, F. Calculation of Solvent Shifts on Electronic G-Tensors with the Conductor-Like Screening Model (COSMO) and its Self-Consistent Generalization to Real Solvents (COSMO-RS). *J. Phys. Chem. A*, **2006**, 110, 2235–2245.
7. Ganyushin, D.; Gilka, N.; Taylor, P. R.; Marian, C. M.; Neese, F. The resolution of the identity approximation for calculations of spin-spin contribution to zero-field splitting parameters. *J. Chem. Phys.*, **2010**, 132, 144111.
8. Duboc, C.; Ganyushin, D.; Sivalingam, K.; Collomb, M. N.; Neese, F. Systematic Theoretical Study of the Zero-Field Splitting in Coordination Complexes of Mn(III). Density Functional Theory versus Multireference Wave Function Approaches. *J. Phys. Chem. A*, **2010**, 114, 10750–10758.
9. Neese, F. First principles approach to Spin-Hamiltonian Parameters. In Misra, S. K., editor, *Multifrequency EPR: Theory and Applications*, pages 297–326. Wiley-VCH, **2009**.
10. Pantazis, D. A.; Orio, M.; Petrenko, T.; Messinger, J.; Lubitz, W.; Neese, F. A new quantum chemical approach to the magnetic properties of oligonuclear transition metal clusters: Application to a model for the tetranuclear manganese cluster of Photosystem II. *Chem. Eur. J.*, **2009**, 15 (20), 5108–5123.
11. Riplinger, C.; Kao, J. P. Y.; Rosen, G. M.; Kathirvelu, V.; Eaton, G. R.; Eaton, S. S.; Kutateladze, A.; Neese, F. Interaction of Radical Pairs Through-Bond and Through-Space: Scope and Limitations of the Point-Dipole Approximation in Electron Paramagnetic Resonance Spectroscopy. *J. Am. Chem. Soc.*, **2009**, 131, 10092–10106.
12. Cirera, J.; Ruiz, E.; Alvarez, S.; Neese, F.; Kortus, J. How to Build Molecules with Large Magnetic Anisotropy. *Chem. Eur. J.*, **2009**, 15 (16), 4078–4087.
13. Neese, F. Spin Hamiltonian Parameters from First Principle Calculations: Theory and Application. In Hanson, G.; Berliner, L., editors, *Biological Magnetic Resonance*, pages 175–232. **2008**.
14. Kossmann, S.; Kirchner, B.; Neese, F. Performance of modern density functional theory for the prediction of hyperfine structure: meta-GGA and double hybrid functionals. *Molec. Phys.*, **2007**, 105, 2049–2071.
15. Neese, F.; Wolf, A.; Reiher, M.; Fleig, T.; Hess, B. A. Higher Order Douglas-Kroll Calculation of Electric Field Gradients. *J. Chem. Phys.*, **2005**, 122, 204107.
16. Ray, K.; Begum, A.; Weyhermüller, T.; Piligkos, S.; van Slageren, J.; Neese, F.; Wieghardt, K. The Electronic Structure of the Isoelectronic, Square Planar Complexes $[Fe^{II}(L)_2]^{2-}$ and $[Co^{III}(L^{Bu})_2]^-$ (L^{2-} and $(L^{Bu})^{2-}$ = benzene-1,2-dithiolates): an Experimental and Density Functional Theoretical Study. *J. Am. Chem. Soc.*, **2005**, 127, 4403–4415. DOI: 10.1021/ja042803i.
17. Neese, Frank. Metal and Ligand Hyperfine Couplings in Transition Metal Complexes: The Effect of Spin–Orbit Coupling as Studied by Coupled Perturbed Kohn–Sham Theory. *J. Chem. Phys.*, **2003**, 118 (9), 3939–3948. DOI: 10.1063/1.1540619.
18. Neese, F. Prediction of Electron Paramagnetic Resonance g-values by Coupled Perturbed Hartree-Fock and Kohn-Sham Theory. *J. Chem. Phys.*, **2001**, 115, 11080–11096.
19. Neese, F. Theoretical Study of Ligand Superhyperfine Structure. Application to Cu(II) Complexes. *J. Phys. Chem. A*, **2001**, 105, 4290–4299.

B.3.15 Dispersion Corrections to DFT

1. Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. *J. Chem. Phys.*, **2010**, 132, 154104.
2. Grimme, S.; Ehrlich, S.; Goerigk, L. *J. Comput. Chem.*, **2011**, 32, 1456.
3. Grimme, S. Accurate description of van der Waals complexes by density functional theory including empirical corrections. *J. Comput. Chem.*, **2004**, 25, 1463. DOI: 10.1002/jcc.20078.
4. Wittmann, Lukas; Gordiy, Igor; Friede, Marvin; Helmich-Paris, Benjamin; Grimme, Stefan; Hansen, Andreas; Bursch, Markus. Extension of the D3 and D4 London dispersion corrections to the full actinides series. *Phys. Chem. Chem. Phys.*, **2024**, 26, 21379–21394. DOI: 10.1039/D4CP01514B.
5. Grimme, S. Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction. *J. Comput. Chem.*, **2006**, 27, 1787. DOI: 10.1002/jcc.20495.

B.3.16 DLPNO

1. Pinski, P.; Riplinger, C.; Valeev, E. F.; Neese, Frank. *J. Chem. Phys.*, **2015**, 143, 034108.
2. Pavošević, F.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E.F. SparseMaps – A systematic infrastructure for reduced-scaling electronic structure methods. IV. Linear-scaling second-order explicitly correlated energy with pair natural orbitals. *J. Chem. Phys.*, **2016**, 144, 144109.
3. Pinski, Peter; Neese, Frank. Communication: Exact Analytical Derivatives for the Domain-Based Local Pair Natural Orbital MP2 Method (DLPNO-MP2). *J. Chem. Phys.*, **2018**, 148, 031101. DOI: 10.1063/1.5011204.
4. Pinski, Peter; Neese, Frank. Analytical Gradient for the Domain-Based Local Pair Natural Orbital Second Order Møller-Plesset Perturbation Theory Method (DLPNO-MP2). *J. Chem. Phys.*, **2019**, 150, 164102.
5. Sparta, Manuel; Retegan, Marius; Pinski, Peter; Riplinger, Christoph; Becker, Ute; Neese, Frank. Multilevel Approaches within the Local Pair Natural Orbital Framework. *J. Chem. Theory Comput.*, **2017**, 13 (7), 3198–3207. DOI: 10.1021/acs.jctc.7b00260.
6. Riplinger, C.; Pinski, P.; Becker, U.; Valeev, E. F.; Neese, Frank. *J. Chem. Phys.*, **2016**, 144, 024109.
7. Datta, D.; Kossmann, S.; Neese, F. Analytic energy derivatives for the calculation of the first-order molecular properties using the domain-based local pair-natural orbital coupled-cluster theory. *J. Chem. Phys.*, **2016**, 175, 114101. DOI: 10.1063/1.4962369.
8. Liakos, Dimitrios G.; Sparta, Manuel; Kesharwani, Manoj K.; Martin, Jan M. L.; Neese, Frank. Exploring the Accuracy Limits of Local Pair Natural Orbital Coupled-Cluster Theory. *J. Chem. Theory Comput.*, **2015**, 11 (4), 1525–1539. DOI: 10.1021/acs.jctc.5b00078.
9. Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. SparseMaps—A Systematic Infrastructure for Reduced-Scaling Electronic Structure Methods. III. Linear-Scaling Multireference Domain-Based Pair Natural Orbital N-Electron Valence Perturbation Theory. *J. Chem. Phys.*, **2016**, 144 (9), 094111. DOI: 10.1063/1.4942769.
10. Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.
11. Dutta, Achintya Kumar; Neese, Frank; Izsák, Róbert. Towards a Pair Natural Orbital Coupled Cluster Method for Excited States. *J. Chem. Phys.*, **2016**, 145 (3), 034102.
12. Schneider, W.; Bistoni, G.; Sparta., M.; Riplinger, C.; Saitow, M.; Auer, A.; Neese, F. Decomposition of Intermolecular Interaction Energies within the Local Pair Natural Orbital Coupled Cluster Framework. *J. Chem. Theory Comput.*, **2016**, 12 (10), 4778–4792. DOI: 10.1021/acs.jctc.6b00523.
13. Kubas, A.; Berger, D.; Oberhofer, H.; Maganas, D.; Reuter, K.; Neese, F. Surface Adsorption Energetics Studied with "Gold Standard" Wave Function-Based Ab Initio Methods: Small-Molecule Binding to TiO2(110). *J. Phys. Chem. Lett.*, **2016**, 7, 4207–4212.

14. Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. Extrapolation to the Limit of a Complete Pair Natural Orbital Space in Local Coupled-Cluster Calculations. *J. Chem. Theory Comput.*, **2020**, 16 (10), 6142–6149. DOI: 10.1021/acs.jctc.0c00344.

15. Guo, Yang; Riplinger, Christoph; Liakos, Dimitrios G.; Becker, Ute; Saitow, Masaaki; Neese, Frank. Linear scaling perturbative triples correction approximations for open-shell domain-based local pair natural orbital coupled cluster singles and doubles theory [DLPNO-CCSD(T/T)]. *J. Chem. Phys.*, **2020**, 152 (2), 024116. DOI: 10.1063/1.5127550.

16. Liakos, D. G.; Guo, Y.; Neese, F. Comprehensive Benchmark Results for the Domain Based Local Pair Natural Orbital Coupled Cluster Method (DLPNO-CCSD(T)) for Closed- and Open-Shell Systems. *J. Phys. Chem. A*, **2020**, 124 (1), 90–100. DOI: 10.1021/acs.jpca.9b05734.

17. Pavošević, Fabijan; Peng, Chong; Pinski, Peter; Riplinger, Christoph; Neese, Frank; Valeev, Edward F. SparseMaps—A systematic infrastructure for reduced scaling electronic structure methods. V. Linear scaling explicitly correlated coupled-cluster method with pair natural orbitals. *J. Chem. Phys.*, **2017**, 146 (17), 174108.

18. Mondal, B.; Neese, F.; Ye, S. F. Control in the Rate-Determining Step Provides a Promising Strategy To Develop New Catalysts for CO2 Hydrogenation: A Local Pair Natural Orbital Coupled Cluster Theory Study. *Inorg. Chem.*, **2015**, 54, 7192–7198.

19. Stoychev, G. L.; Auer, A. A.; Gauss, J.; Neese, F. DLPNO-MP2 second derivatives for the computation of polarizabilities and NMR shieldings. *J. Chem. Phys.*, **2021**, 154, 164110.

20. Isegawa, M.; Neese, F.; Pantazis, D. A. Ionization Energies and Aqueous Redox Potentials of Organic Molecules: Comparison of DFT, Correlated ab Initio Theory and Pair Natural Orbital Approaches. *J. Chem. Theory Comput.*, **2016**, 12, 2272–2284.

21. Liakos, D. G.; Neese, F. Domain Based Pair Natural Orbital Coupled Cluster Studies on Linear and Folded Alkane Chains. *J. Chem. Theory Comput.*, **2015**, 11, 2137–2143.

22. Liakos, D. G.; Neese, F. Is It Possible To Obtain Coupled Cluster Quality Energies at near Density Functional Theory Cost? Domain-Based Local Pair Natural Orbital Coupled Cluster vs Modern Density Functional Theory. *J. Chem. Theory Comput.*, **2015**, 11, 4054–4063.

23. Demel, O.; Pittner, J.; Neese, F. A Local Pair Natural Orbital-Based Multireference Mukherjee's Coupled Cluster Method. *J. Chem. Theory Comput.*, **2015**, 11, 3104–3114.

24. Ni, Z. G.; Guo, Y.; Neese, F.; Li, W.; Li, S. H. Cluster-in-Molecule Local Correlation Method with an Accurate Distant Pair Correction for Large Systems. *J. Chem. Theo. Comp.*, **2021**, 17, 756–766.

B.3.17 Density Functionals

1. Bursch, Markus; Neugebauer, Hagen; Ehlert, Sebastian; Grimme, Stefan. Dispersion corrected r^2SCAN based global hybrid functionals: r^2SCANh, r^2SCAN0, and r^2SCAN50. *J. Chem. Phys.*, **2022**, 156 (13), 134105. DOI: 10.1063/5.0086040.

2. Wittmann, Lukas; Neugebauer, Hagen; Grimme, Stefan; Bursch, Markus. Dispersion-corrected r^2SCAN based double-hybrid functionals. *J. Chem. Phys.*, **2023**, 159 (22), 224103. DOI: 10.1063/5.0174988.

3. Neese, F.; Schwabe, T.; Grimme, S. *J. Chem. Phys.*, **2007**, 126, 124115.

4. Ehlert, Sebastian; Huniar, Uwe; Ning, Jinliang; Furness, James W.; Sun, Jianwei; Kaplan, Aaron D.; Perdew, John P.; Brandenburg, Jan Gerit. r^2SCAN-D4: Dispersion Corrected Meta-Generalized Gradient Approximation for General Chemical Applications. *J. Chem. Phys.*, **2021**, 154 (6), 061101. arXiv:10.1063/5.0041008, DOI: 10.1063/5.0041008.

5. Grimme, S.; Neese, F. *J. Phys. Chem.*, **2007**, 127, 154116.

6. Kossmann, S.; Kirchner, B.; Neese, F. Performance of modern density functional theory for the prediction of hyperfine structure: meta-GGA and double hybrid functionals. *Molec. Phys.*, **2007**, 105, 2049–2071.

B.3.18 Composite Methods

1. Müller, Marcel; Hansen, Andreas; Grimme, Stefan. ωB97X-3c: A composite range-separated hybrid DFT method with a molecule-optimized polarized valence double-ζ basis set. *J. Chem. Phys.*, **2023**, 158 (1), 014103. DOI: 10.1063/5.0133026.
2. Grimme, Stefan; Hansen, Andreas; Ehlert, Sebastian; Mewes, Jan-Michael. r2SCAN-3c: A "Swiss Army Knife" Composite Electronic-Structure Method. *J. Chem. Phys.*, **2021**, 154 (6), 064103. arXiv:10.1063/5.0040021, DOI: 10.1063/5.0040021.
3. Grimme, S.; Brandenburg, J. G.; Bannwarth, C.; Hansen, A. Consistent structures and interactions by density functional theory with small atomic orbital basis sets. *J. Chem. Phys.*, **2015**, 143, 054107. DOI: 10.1063/1.4927476.
4. Pracht, Philipp; Grant, David F.; Grimme, Stefan. Comprehensive Assessment of GFN Tight-Binding and Composite Density Functional Theory Methods for Calculating Gas-Phase Infrared Spectra. *J. Chem. Theory Comput.*, **2020**, 16 (11), 7044–7060. DOI: 10.1021/acs.jctc.0c00877.
5. Sure, R.; Grimme, S. *J. Comput. Chem.*, **2013**, 34, 1672–1685.
6. Brandenburg, J. G.; Bannwarth, C.; Hansen, A.; Grimme, S. B97-3c: A Revised Low-Cost Variant of the B97-D Density Functional Method. *J. Chem. Phys.*, **2018**, 148 (6), 064104.

B.3.19 Extended Tight-Binding Methods

1. Bannwarth, Christoph; Ehlert, Sebastian; Grimme, Stefan. GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1652–1671. DOI: 10.1021/acs.jctc.8b01176.
2. Grimme, Stefan; Bannwarth, Christoph; Shushkov, Philip. A Robust and Accurate Tight-Binding Quantum Chemical Method for Structures, Vibrational Frequencies, and Noncovalent Interactions of Large Molecular Systems Parametrized for All Spd-Block Elements (z = 1-86). *J. Chem. Theory Comput.*, **2017**, 13 (5), 1989–2009. DOI: 10.1021/acs.jctc.7b00118.
3. Pracht, Philipp; Caldeweyher, Eike; Ehlert, Sebastian; Grimme, Stefan. A Robust Non-Self-Consistent Tight-Binding Quantum Chemistry Method for Large Molecules. ChemRxiv, **2019**. DOI: 10.26434/chemrxiv.8326202.v1.
4. Ehlert, Sebastian; Stahn, Marcel; Spicher, Sebastian; Grimme, Stefan. Robust and Efficient Implicit Solvation Model for Fast Semiempirical Methods. *J. Chem. Theory Comput.*, **2021**, 17 (7), 4250–4261. DOI: 10.1021/acs.jctc.1c00471.
5. Plett, Christoph; Katbashev, Abylay; Ehlert, Sebastian; Grimme, Stefan; Bursch, Markus. ONIOM meets xtb: efficient, accurate, and robust multi-layer simulations across the periodic table. *Phys. Chem. Chem. Phys.*, **2023**, 25, 17860–17868. DOI: 10.1039/D3CP02178E.
6. Bursch, Markus; Hansen, Andreas; Grimme, Stefan. Fast and Reasonable Geometry Optimization of Lanthanoid Complexes with an Extended Tight Binding Quantum Chemical Method. *Inorg. Chem.*, **2017**, 56 (20), 12485–12491. DOI: 10.1021/acs.inorgchem.7b01950.
7. Bursch, Markus; Neugebauer, Hagen; Grimme, Stefan. Structure Optimisation of Large Transition-Metal Complexes with Extended Tight-Binding Methods. *Angew. Chem. Int. Ed.*, **2019**, 58 (32), 11078–11087. DOI: 10.1002/anie.201904021.
8. Neugebauer, Hagen; Bohle, Fabian; Bursch, Markus; Hansen, Andreas; Grimme, Stefan. Benchmark Study of Electrochemical Redox Potentials Calculated with Semiempirical and DFT Methods. *J. Phys. Chem. A*, **2020**, 124 (35), 7166–7176. DOI: 10.1021/acs.jpca.0c05052.
9. Bannwarth, Christoph; Caldeweyher, Eike; Ehlert, Sebastian; Hansen, Andreas; Pracht, Philipp; Seibert, Jakob; Spicher, Sebastian; Grimme, Stefan. Extended tight-binding quantum chemistry methods. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2021**, 11 (2), e1493. DOI: 10.1002/wcms.1493.

B.3.20 Excited States Methods and Resonance Raman Spectra

1. Schapiro, Igor; Sivalingam, Kantharuban; Neese, Frank. Assessment of N-Electron Valence State Perturbation Theory for Vertical Excitation Energies. *J. Chem. Theory Comput.*, **2013**, 9 (8), 3567–3580. DOI: 10.1021/ct400136y.
2. Roemelt, M.; Neese, F. Excited States of Large Open-Shell Molecules: An Efficient, General, and Spin-Adapted Approach Based on a Restricted Open-Shell Ground State Wave function. *J. Phys. Chem. A*, **2013**, 117, 3069–3082. DOI: 10.1021/jp3126126.
3. Petrenko, T.; Kossmann, S.; Neese, F. *J. Chem. Phys.*, **2011**, 134, 054116.
4. Grimme, S.; Neese, F. *J. Phys. Chem.*, **2007**, 127, 154116.
5. Petrenko, T.; Ray, K.; Wieghardt, K.; Neese, F. Vibrational Markers for the Open-Shell Character of Metal bis-Dithiolenes: An Infrared, resonance Raman and Quantum Chemical Study. *J. Am. Chem. Soc.*, **2006**, 128, 4422–4436.
6. Neese, F.; Olbrich, G. Efficient use of the Resolution of the Identity Approximation in Time-Dependent Density Functional Calculations with Hybrid Functionals. *Chem. Phys. Lett.*, **2002**, 362, 170–178.

B.3.21 F12

1. Pavošević, F.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E.F. SparseMaps – A systematic infrastructure for reduced-scaling electronic structure methods. IV. Linear-scaling second-order explicitly correlated energy with pair natural orbitals. *J. Chem. Phys.*, **2016**, 144, 144109.
2. Liakos, Dimitrios G.; Izsák, Róbert; Valeev, Edward F.; Neese, Frank. What is the most efficient way to reach the canonical MP2 basis set limit? *Mol. Phys.*, **2013**, 111 (19-20), 2653–2662. DOI: 10.1080/00268976.2013.811812.
3. Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. Explicitly Correlated N-Electron Valence State Perturbation Theory (NEVPT2-F12). *J. Chem. Phys.*, **2017**, 147 (6), 064110. DOI: 10.1063/1.4996560.
4. Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.

B.3.22 FOD analysis and FOD plots

1. Grimme, S.; Hansen, A. A Practicable Real-Space Measure and Visualization of Static Electron-Correlation Effects. *Angew. Chem. Int. Ed.*, **2015**, 54, 12308–12313.

B.3.23 Gaussian Charge Scheme (C-PCM) Implementation

1. Garcia-Ratés, M.; Neese, F. Effect of the Solute Cavity on the Solvation Energy and its Derivatives within the Framework of the Gaussian Charge Scheme. *J. Comput. Chem.*, **2020**, 41 (9), 922–939. DOI: 10.1002/jcc.26139.

B.3.24 gCP correction to HF and DFT

1. Kruse, Holger; Grimme, Stefan. A geometrical correction for the inter- and intra-molecular basis set superposition error in Hartree-Fock and density functional theory calculations for large systems. *J. Chem. Phys.*, **2012**, 136 (15), 154101. DOI: 10.1063/1.3700154.

B.3.25 Internally Contracted Multireference CI

1. Sivalingam, K.; Krupicka, M.; Auer, A. A.; Neese, F. Comparison of fully internally and strongly contracted multireference configuration interaction procedures. *J. Chem. Phys.*, **2016**, 145, 054104. DOI: 10.1063/1.4950161.

B.3.26 Magnetic Circular Dichroism Spectra

1. Ganyushin, D.; Neese, F. *J. Chem. Phys.*, **2008**, 128, 114117.
2. Westphal, A.; Broda, H.; Kurz, P.; Neese, F.; Tuczek, F. Magnetic Circular Dichroism Spectrum of the Molybdenum(V) Complex (Mo(O)Cl_3dppe): C-Term Signs and Intensities for Multideterminant Excited Doublet States. *Inorg. Chem.*, **2012**, 51, 5748–5763.
3. van Slageren, J.; Piligkos, S.; Neese, F. Magnetic circular dichroism spectroscopy on the Cr(8) antiferromagnetic ring. *Dalton Trans.*, **2010**, 39, 4999–5004.
4. Sundararajan, M.; Ganyushin, D.; Ye, S.; Neese, F. Multireference \it ab initio studies of Zero-Field Splitting and Magnetic Circular Dichroism Spectra of Tetrahedral Co(II) Complexes. *Dalton Trans.*, **2009**, 30, 6021–6036.
5. Piligkos, S.; Slep, L.; Weyhermüller, T.; Chaudhuri, P.; Bill, E.; Neese, F. Magnetic Circular Dichroism Spectroscopy of weakly exchange coupled dimers. A model study. *Coord. Chem. Rev.*, **2009**, 253, 2352–2362.
6. Neese, F.; Solomon, E. I. MCD C-term Signs, Saturation Behavior and Determination of Band Polarizations in Randomly Oriented Systems with Spin S ? 1/2. Applications to S=1/2 and S=5/2. *Inorg. Chem.*, **1999**, 38, 1847–1865.

B.3.27 MCD

1. Ye, S. F.; Kupper, C.; Meyer, S.; Andris, E.; Navratil, R.; Krahe, O.; Mondal, B.; Atanasov, M.; Bill, E.; Roithova, J.; Meyer, F.; Neese, F. Magnetic Circular Dichroism Evidence for an Unusual Electronic Structure of a Tetracarbene-Oxoiron(IV) Complex. *J. Am. Chem. Soc.*, **2016**, 138, 14312–14325.
2. Ye, S. F.; Xue, G. Q.; Krivokapic, I.; Petrenko, T.; Bill, E.; Que, L.; Neese, F. Magnetic circular dichroism and computational study of mononuclear and dinuclear iron(IV) complexes. *Chem. Sci.*, **2015**, 6, 2909–2921.

B.3.28 MDCI

1. Dutta, Achintya Kumar; Neese, Frank; Izsák, Róbert. Speeding up Equation of Motion Coupled Cluster Theory with the Chain of Spheres Approximation. *J. Chem. Phys.*, **2016**, 144 (3), 034102.
2. Veis, Libor; Antalík, Andrej; Brabec, Jiri; Neese, Frank; Legeza, Ors; Pittner, Jiri. Coupled cluster method with single and double excitations tailored by matrix product state wave functions. *J. Phys. Chem. Lett.*, **2016**, 7 (20), 4072–4078.

B.3.29 Mössbauer Spectroscopy

1. Neese, F. *Inorg. Chim. Acta*, **2002**, 337C, 181–192.
2. Römelt, M.; Ye, S.; Neese, F. *Inorg. Chem.*, **2009**, 48, 784.
3. Datta, D.; Saitow, M.; Sandhofer, B.; Neese, F. Fe-57 Mossbauer parameters from domain based local pair-natural orbital coupled-cluster theory. *J. Chem. Phys.*, **2020**, 153, 064104.
4. Sinnecker, S.; Slep, L.; Bill, E.; Neese, F. Performance of Nonrelativistic and Quasirelativistic Hybrid DFT for the Prediction of Electric and Magnetic Hyperfine Parameters in ^{57}Fe Mössbauer Spectra. *Inorg. Chem.*, **2005**, 44, 2245–2254.

B.3.30 Multireference CI Module and its application to EPR properties and optical spectra

1. Neese, F. Importance of Direct Spin-Spin Coupling and Spin-Flip Excitations for the Zero-Field Splittings of Transition Metal Complexes: A Case Study. *J. Am. Chem. Soc.*, **2006**, 128, 10213.
2. Atanasov, Mihail; Ganyushin, Dmitry; Pantazis, Dimitrios A.; Sivalingam, Kantharuban; Neese, Frank. Detailed Ab Initio First-Principles Study of the Magnetic Anisotropy in a Family of Trigonal Pyramidal Iron(II) Pyrrolide Complexes. *Inorg. Chem.*, **2011**, 50 (16), 7460–7477. DOI: 10.1021/ic200196k.
3. Neese, Frank; Petrenko, Taras; Ganyushin, Dmitry; Olbrich, Gottfried. Advanced Aspects of Ab Initio Theoretical Optical Spectroscopy of Transition Metal Complexes: Multiplets, Spin-Orbit Coupling and Resonance Raman Intensities. *Coordin. Chem. Rev.*, **2007**, 251 (3-4), 288–327. DOI: 10.1016/j.ccr.2006.05.019.
4. Neese, F. *Chem. Phys. Lett.*, **2003**, 380, 721–728.
5. Ganyushin, D.; Neese, F. *J. Chem. Phys.*, **2008**, 128, 114117.
6. Nooijen, Marcel; Demel, Ondrej; Datta, Dipayan; Kong, Liguo; Shamasundar, K. R.; Lotrich, V.; Huntington, Lee M.; Neese, Frank. Communication: Multireference Equation of Motion Coupled Cluster: A Transform and Diagonalize Approach to Electronic Structure. *J. Chem. Phys.*, **2014**, 140 (8), 081102.
7. Neese, F.; Solomon, E. I. *Inorg. Chem.*, **1998**, 37, 6568–6582.
8. Petrenko, Taras; Neese, Frank. Analysis and prediction of absorption band shapes, fluorescence band shapes, resonance Raman intensities, and excitation profiles using the time-dependent theory of electronic spectroscopy. *J. Chem. Phys.*, **2007**, 127 (16), 164319. DOI: 10.1063/1.2770706.
9. Ganyushin, D.; Neese, F. *J. Chem. Phys.*, **2013**, 138, 104113.
10. Duboc, C.; Ganyushin, D.; Sivalingam, K.; Collomb, M. N.; Neese, F. Systematic Theoretical Study of the Zero-Field Splitting in Coordination Complexes of Mn(III). Density Functional Theory versus Multireference Wave Function Approaches. *J. Phys. Chem. A*, **2010**, 114, 10750–10758.
11. Demel, O.; Pittner, J.; Neese, F. A Local Pair Natural Orbital-Based Multireference Mukherjee's Coupled Cluster Method. *J. Chem. Theory Comput.*, **2015**, 11, 3104–3114.
12. Sundararajan, M.; Ganyushin, D.; Ye, S.; Neese, F. Multireference \it ab initio studies of Zero-Field Splitting and Magnetic Circular Dichroism Spectra of Tetrahedral Co(II) Complexes. *Dalton Trans.*, **2009**, 30, 6021–6036.
13. Lang, L.; Sivalingam, K.; Neese, F. The combination of multipartitioning of the Hamiltonian with canonical Van Vleck perturbation theory leads to a Hermitian variant of quasidegenerate N-electron valence perturbation theory. *J. Chem. Phys.*, **2020**, 152, 034106.
14. Atanasov, M.; Zadrozny, J. M.; Long, J. R.; Neese, F. A theoretical analysis of chemical bonding, vibronic coupling, and magnetic anisotropy in linear iron(II) complexes with single-molecule magnet behavior. *Chem. Sci.*, **2013**, 4, 139–156.
15. Neese, F.; Pantazis, D. A. What is not required to make a single molecule magnet. *Faraday Discuss.*, **2011**, 148, 229–238.

16. Neese, F. Analytic Derivative Calculation of Electronic g-Tensors based on Multireference Configuration Interaction Wavefunctions. *Mol. Phys.*, **2007**, 105, 2507–2514.

17. Neese, Frank. Theoretical spectroscopy of model-nonheme [Fe(IV)OL5]2+ complexes in their lowest triplet and quintet states using multireference ab initio and density functional theory methods. *J. Inorg. Biochem.*, **2006**, 100 (4), 716–726. High-valent iron intermediates in biology. DOI: 10.1016/j.jinorgbio.2006.01.020.

18. Ganyushin, D.; Neese, F. First Principle Calculation of Zero-Field Splittings. *J. Chem. Phys.*, **2006**, 125, 024103.

19. Ray, K.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Electronic Structure of Square-Planar Bis(benzene-1,2-dithiolate)metal Complexes $[M(L)_2]^z$ (z=2-,1-,0; M = Ni, Pd, Pt, Cu, Au): An experimental, Density Functional and Correlated \it ab initio Study. *Inorg. Chem.*, **2005**, 44, 5345–5360.

20. Schöneboom, J.; Neese, F.; Thiel, W. Towards Identification of the Compound I Reactive Intermediate in Cytochrome P450 Chemistry: A QM/MM Study of its EPR and Mössbauer Parameters. *J. Am. Chem. Soc.*, **2005**, 127, 5840–5853.

21. Wanko, M.; Hoffmann, M.; Strodel, P.; Thiel, W.; Neese, F.; Frauenheim, T.; Elstner, M. Calculating Absorption Shifts for Retinal Proteins: Computational Challenges. *J. Phys. Chem. B*, **2005**, 109, 3606–3615.

22. Neese, F. Sum Over States Based Multireference \it ab initio Calculation of EPR Spin Hamiltonian Parameters for Transition Metal Complexes. A Case Study. *Mag. Res. Chem.*, **2004**, 42, S187–S198.

23. Neese, F. A Spectroscopy Oriented Configuration Interaction Procedure. *J. Chem. Phys.*, **2003**, 119, 9428–9443.

24. Neese, F. Configuration Interaction Calculation of Electronic g-Tensors in Transition Metal Complexes. *Int. J. Quant. Chem.*, **2001**, 83, 104–114.

25. Chalupský, J.; Neese, F.; Solomon, E. I.; Ryde, U.; Rulíšek, L. Identification of intermediates in the reaction cycle of multicopper oxidases by quantum chemical calculations of spectroscopic parameters. *Inorg. Chem.*, **2006**, 45, 11051–11059.

B.3.31 NRVS

1. Ogata, H.; Kramer, T.; Wang, H. X.; Schilter, D.; Pelmenschikov, V.; van Gastel, M.; Neese, F.; Rauchfuss, T. B.; Gee, L. B.; Scott, A. D.; Yoda, Y.; Tanaka, Y.; Lubitz, W.; Cramer, S. P. Hydride bridge in NiFe-hydrogenase observed by nuclear resonance vibrational spectroscopy. *Nat. Commun.*, **2015**, 6, 7890.

B.3.32 Nuclear Resonance Vibrational Spectra

1. Petrenko, Taras; DeBeer-George, Serena; Aliaga-Alcalde, Núria; Bill, Eckhard; Mienert, Bernd; Xiao, Yuming; Guo, YiSong; Sturhahn, Wolfgang; Cramer, Stephen P.; Wieghardt, Karl; Neese, Frank. Characterization of a Genuine Iron(V)-Nitrido Species by Nuclear Resonant Vibrational Spectroscopy Coupled to Density Functional Calculations. *J. Am. Chem. Soc.*, **2007**, 129, 11053–11060.

2. Petrenko, T.; Sturhahn, W.; Neese, F. First principles calculation of Nuclear Resonance Vibrational Spectra. *Hyperfine Interact.*, **2008**, 175, 165–174.

B.3.33 Orbital Optimized MP2

1. Neese, F.; Schwabe, T.; Kossmann, S.; Schirmer, B.; Grimme, S. *J. Chem. Theory Comput.*, **2009**, 5, 3060.
2. Shirazi, R. G.; Pantazis, D. A.; Neese, F. Performance of density functional theory and orbital-optimised second-order perturbation theory methods for geometries and singlet-triplet state splittings of aryl-carbenes. *Mol. Phys.*, **2020**, 118, e1750456.
3. Sandhoefer, B.; Kossmann, S.; Neese, F. Derivation and assessment of relativistic hyperfine-coupling tensors on the basis of orbital-optimized second-order Møller-Plesset perturbation theory and the second-order Douglas–Kroll–Hess transformation. *J. Chem. Phys.*, **2013**, 138, 104102.
4. Kossmann, S.; Neese, F. Correlated ab Initio Spin Densities for Larger Molecules: Orbital-Optimized Spin-Component-Scaled MP2 Method. *J. Phys. Chem. A*, **2010**, 114, 11768–11781.

B.3.34 Pair Natural Orbital Local Correlation Methods

1. Kollmar, Christian; Neese, Frank. The relationship between double excitation amplitudes and Z vector components in some post-Hartree-Fock correlation methods. *J. Chem. Phys.*, **2011**, 135 (6), 064103. DOI: 10.1063/1.3618720.
2. Kollmar, Christian; Neese, Frank. An orbital-invariant and strictly size extensive post-Hartree-Fock correlation functional. *J. Chem. Phys.*, **2011**, 135 (8), 084102. DOI: 10.1063/1.3624567.
3. Huntington, Lee M. J.; Hansen, Andreas; Neese, Frank; Nooijen, Marcel. Accurate Thermochemistry from a Parameterized Coupled-Cluster Singles and Doubles Model and a Local Pair Natural Orbital Based Implementation for Applications to Larger Systems. *J. Chem. Phys.*, **2012**, 136, 064101. DOI: 10.1063/1.3682325.
4. Neese, F.; Hansen, A.; Liakos, D. G. *J. Chem. Phys.*, **2009**, 131, 064103.
5. Hansen, A.; Liakos, D. G.; Neese, F. *J. Chem. Phys.*, **2011**, 135, 214102.
6. Liakos, D. G.; Neese, F. Improved correlation energy extrapolation schemes based on local pair natural orbital methods. *J. Phys. Chem. A*, **2012**, 116 (19), 4801–4816. DOI: 10.1021/jp300997x.
7. Riplinger, C.; Neese, F. *J. Chem. Phys.*, **2013**, 138, 034106.
8. Neese, F.; Wennmohs, F.; Hansen, A. *J. Chem. Phys.*, **2009**, 130, 114108.
9. Riplinger, C.; Sandhoefer, B.; Hansen, A.; Neese, F. *J. Chem. Phys.*, **2013**, 139, 134101.
10. Izsák, R.; Hansen, A.; Neese, F. The resolution of identity and chain of spheres approximations for the LPNO-CCSD singles Fock term. *Mol. Phys.*, **2012**, 110, 2413–2417.
11. Liakos, D. G.; Hansen, A.; Neese, F. Weak Molecular Interactions Studied with Parallel Implementations of the Local Pair Natural Orbital Coupled Pair and Coupled Cluster Methods. *J. Chem. Theory Comput.*, **2011**, 7, 76–87.

B.3.35 PBEh-3c method

1. Grimme, S.; Brandenburg, J. G.; Bannwarth, C.; Hansen, A. Consistent structures and interactions by density functional theory with small atomic orbital basis sets. *J. Chem. Phys.*, **2015**, 143, 054107. DOI: 10.1063/1.4927476.

B.3.36 QM/MM

1. Rokhsana, D.; Large, T. A. G.; Dienst, M. C.; Retegan, M.; Neese, F. A realistic in silico model for structure/function studies of molybdenum-copper CO dehydrogenase. *Inorg. Chem.*, **2016**, 21, 491–499.
2. Retegan, M.; Krewald, V.; Mamedov, F.; Neese, F.; Lubitz, W.; Cox, N.; Pantazis, D. A. A five-coordinate Mn(IV) intermediate in biological water oxidation: spectroscopic signature and a pivot mechanism for water binding. *Chem. Sci.*, **2016**, 7, 72–84.
3. Sundararajan, M.; Neese, F. Distal Histidine Modulates the Unusual O-Binding of Nitrite to Myoglobin: Evidence from the Quantum Chemical Analysis of EPR Parameters. *Inorg. Chem.*, **2015**, 54, 7209–7217.

B.3.37 QM/MM calculations with ORCA

1. Schulz, C. E.; van Gastel, M.; Pantazis, D. A.; Neese, F. Converged Structural and Spectroscopic Properties for Refined QM/MM Models of Azurin. *Inorg. Chem.*, **2021**, 60 (10), 7399–7412. DOI: 10.1021/acs.inorgchem.1c00640.
2. Schulz, C. E.; Castillo, R. G.; Pantazis, D. A.; DeBeer, S.; Neese, F. Structure-Spectroscopy Correlations for Intermediate q of Soluble Methane Monooxygenase: Insights from QM/MM Calculations. *J. Am. Chem. Soc.*, **2021**, 143 (17), 6560–6577. DOI: 10.1021/jacs.1c01180.
3. Schöneboom, J.; Neese, F.; Thiel, W. Towards Identification of the Compound I Reactive Intermediate in Cytochrome P450 Chemistry: A QM/MM Study of its EPR and Mössbauer Parameters. *J. Am. Chem. Soc.*, **2005**, 127, 5840–5853.
4. Wanko, M.; Hoffmann, M.; Strodel, P.; Thiel, W.; Neese, F.; Frauenheim, T.; Elstner, M. Calculating Absorption Shifts for Retinal Proteins: Computational Challenges. *J. Phys. Chem. B*, **2005**, 109, 3606–3615.
5. Sundararajan, M.; Neese, F. Detailed QM/MM study of the Electron Paramagnetic Resonance Parameters of Nitrosyl Myoglobin. *J. Chem. Theory Comput.*, **2012**, 8, 563–574.
6. Radoul, M.; Sundararajan, M.; Potapov, A.; Riplinger, C.; Neese, F.; Goldfarb, D. Revisiting the nitrosyl complex of myoglobin by high-field pulse EPR spectroscopy and quantum mechanical calculations. *Phys. Chem. Chem. Phys.*, **2010**, 12, 7276–7289.
7. Sundararajan, M.; Riplinger, C.; Orio, M.; Wennmohs, F.; Neese, F. Spectroscopic Properties of Protein-Bound Cofactors: Calculation by Combined Quantum Mechanical/Molecular Mechanical (QM/MM) Approaches. In *Encyclopedia of Inorganic Chemistry*. **2009**.
8. Altun, A.; Kumar, D.; Neese, F.; Thiel, W. Multi-reference Ab Initio QM/MM Study on Intermediates in the Catalytic Cycle of Cytochrome P450cam. *J. Phys. Chem.*, **2008**, 112, 12904–12910.
9. Sinnecker, S.; Neese, F. QM/MM Calculations with DFT for Taking into Account Protein Effects on the EPR and Optical Spectra of Metalloproteins. Plastocyanin as a Case Study. *J. Comp. Chem.*, **2006**, 27, 1463–1475.
10. Riplinger, C.; Neese, F. The Reaction Mechanism of Cytochrome P450 NO Reductase: A Detailed Quantum Mechanics/Molecular Mechanics Study. *ChemPhysChem*, **2011**, 12, 3192–3203.
11. Chalupský, J.; Neese, F.; Solomon, E. I.; Ryde, U.; Rulíšek, L. Identification of intermediates in the reaction cycle of multicopper oxidases by quantum chemical calculations of spectroscopic parameters. *Inorg. Chem.*, **2006**, 45, 11051–11059.

B.3.38 Relativity and SARC Basis Sets

1. Pantazis, D. A.; Chen, X.-Y.; Landis, C. R.; Neese, F. *J. Chem. Theory Comput.*, **2008**, 4, 908–919.
2. Bühl, M.; Reimann, C.; Pantazis, D. A.; Bredow, T.; Neese, F. Geometries of Third-Row Transition-Metal Complexes from Density-Functional Theory. *J. Chem. Theory Comput.*, **2008**, 4, 1449–1459. DOI: 10.1021/ct800172j.
3. Pantazis, D. A.; Neese, F. *J. Chem. Theory Comput.*, **2009**, 5, 2229–2238.
4. Pantazis, D. A.; Neese, F. *J. Chem. Theory Comput.*, **2011**, 7, 677–684.
5. Pantazis, D. A.; Neese, F. *Theor. Chem. Acc.*, **2012**, 131, 1292.
6. Rolfes, Julian D.; Neese, Frank; Pantazis, Dimitrios A. All-Electron Scalar Relativistic Basis Sets for the Elements Rb–Xe. *J. Comput. Chem.*, **2020**, 41, 1842–1849. DOI: 10.1002/jcc.26355.
7. Aravena, D.; Neese, F.; Pantazis, Dimitrios A. Improved Segmented All-Electron Relativistically Contracted Basis Sets for the Lanthanides. *J. Chem. Theory Comput.*, **2016**, 12, 1148–1156. DOI: 10.1021/acs.jctc.5b01048.

B.3.39 Resonance Raman

1. de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Efficient Simulation of Overtones and Combination Bands in Resonant Raman Spectra. *J. Chem. Phys.*, **2019**, 150 (21), 044105. DOI: 10.1063/1.5099247.
2. Maganas, D.; Trunschke, A.; Schlogl, R.; Neese, F. A unified view on heterogeneous and homogeneous catalysts through a combination of spectroscopy and quantum chemistry. *Faraday Discuss.*, **2016**, 188, 181–197.

B.3.40 SOC on TD-DFT

1. de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Predicting Phosphorescence Rates of Light Organic Molecules Using Time-Dependent Density Functional Theory and the Path Integral Approach to Dynamics. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1896–1904. DOI: 10.1021/acs.jctc.8b00841.

B.3.41 SOSCF Method

1. Neese, F. Approximate Second Order Convergence for Spin Unrestricted Wavefunctions. *Chem. Phys. Lett.*, **2000**, 325, 93–98.

B.3.42 The Split-J, Split-RI-J, RIJCOSX and RI-JK methods

1. Neese, F.; Wennmohs, F.; Hansen, A.; Becker, U. *Chem. Phys.*, **2009**, 356, 98–109.
2. Kossmann, Simone; Neese, Frank. Comparison of two efficient approximate Hartee–Fock approaches. *Chem. Phys. Lett.*, **2009**, 481 (4-6), 240–243. DOI: 10.1016/j.cplett.2009.10.007.
3. Izsák, Róbert; Neese, Frank. An Overlap Fitted Chain of Spheres Exchange Method. *J. Chem. Phys.*, **2011**, 135, 144105. DOI: 10.1063/1.3644029.
4. Izsák, R.; Neese, F. Speeding up spin-component-scaled third-order pertubation theory with the chain of spheres approximation: the COSX-SCS-MP3 method. *Mol. Phys.*, **2013**, 111, 1190.
5. Kossmann, S.; Neese, F. Efficient Structure Optimization with Second-Order Many-Body Perturbation Theory: The RIJCOSX-MP2 Method. *J. Chem. Theory Comput.*, **2010**, 6, 2325–2338.
6. Neese, F. An Improvement of the Resolution of the Identity Approximation for the Calculation of the Coulomb Matrix. *J. Comp. Chem.*, **2003**, 24, 1740–1747.

B.3.43 sTDA and sTD-DFT approaches for electronic spectra

1. Grimme, S. A simplified Tamm–Dancoff density functional approach for the electronic excitation spectra of very large molecules. *J. Chem. Phys.*, **2013**, 138, 244104. DOI: 10.1063/1.4811330.
2. Bannwarth, C.; Grimme, S. A simplified time-dependent density functional theory approach for electronic ultraviolet and circular dichroism spectra of very large molecules. *Comp. Theor. Chem.*, **2014**, 1040 –1041, 45–53. DOI: 10.1016/j.comptc.2014.02.023.
3. Risthaus, T.; Hansen, A.; Grimme, S. *Phys. Chem. Chem. Phys.*, **2014**, 16, 14408–14419.

B.3.44 XAS/XES

1. Van Kuiken, B. E.; Hahn, A. W.; Maganas, D.; DeBeer, S. Measuring Spin-Allowed and Spin-Forbidden d-d Excitations in Vanadium Complexes with 2p3d Resonant Inelastic X-ray Scattering. *Inorg. Chem.*, **2016**, 55, 11497–11501.
2. Rees, J. A.; Wandzilak, A.; Maganas, D.; Wurster, N. I. C.; Hugenbruch, S.; Kowalska, J. K.; Pollock, C. J.; Lima, F. A.; Finkelstein, K. D.; DeBeer, S. Experimental and theoretical correlations between vanadium K-edge X-ray absorption and K emission spectra. *J. Biol. Inorg. Chem.*, **2016**, 21, 793–805.
3. Martin-Diaconescu, V.; Chacon, K. N.; Delgado-Jaime, M. U.; Sokaras, D.; Weng, T. C.; DeBeer, S.; Blackburn, N. J. K β Valence to Core X-ray Emission Studies of Cu(I) Binding Proteins with Mixed Methionine - Histidine Coordination. Relevance to the Reactivity of the M- and H-sites of Peptidylglycine Monooxygenase. *Inorg. Chem.*, **2016**, 55, 3431–3439.
4. Kowalska, J. K.; Hahn, A. W.; Albers, A.; Schiewer, C. E.; Bjornsson, R.; Lima, F. A.; Meyer, F.; DeBeer, S. X-ray Absorption and Emission Spectroscopic Studies of L2Fe2S2 (n) Model Complexes: Implications for the Experimental Evaluation of Redox States in Iron-Sulfur Clusters. *Inorg. Chem.*, **2016**, 55, 4485–4497.
5. Rees, J. A.; Martin-Diaconescu, V.; Kovacs, J. A.; DeBeer, S. X-ray Absorption and Emission Study of Dioxygen Activation by a Small-Molecule Manganese Complex. *Inorg. Chem.*, **2015**, 54, 6410–6422.
6. Rees, J. A.; Bjornsson, R.; Schlesier, J.; Sippel, D.; Einsle, O.; DeBeer, S. The Fe-V Cofactor of Vanadium Nitrogenase Contains an Interstitial Carbon Atom. *Angew. Chem. Int. Ed.*, **2015**, 54, 13249–13252.
7. Maganas, D.; Trunschke, A.; Schlogl, R.; Neese, F. A unified view on heterogeneous and homogeneous catalysts through a combination of spectroscopy and quantum chemistry. *Faraday Discuss.*, **2016**, 188, 181–197.

B.3.45 X-Ray Absorption and X-Ray Emission Spectra

1. Maganas, Dimitrios; Kowalska, Joanna K.; Nooijen, Marcel; DeBeer, Serena; Neese, Frank. Comparison of Multireference Ab Initio Wavefunction Methodologies for X-Ray Absorption Edges: A Case Study on $[Fe(II/III)Cl_4]^{2-/1-}$ Molecules. *J. Chem. Phys.*, **2019**, 150 (10), 104106. DOI: 10.1063/1.5051613.
2. Roemelt, M.; Beckwith, M. A.; Duboc, C.; Collomb, M.-N.; Neese, F.; DeBeer, S. Manganese K-Edge X-Ray Absorption Spectroscopy as a Probe of the Metal-Ligand Interactions in Coordination Compounds. *Inorg. Chem.*, **2012**, 51, 680–687.
3. Chandrasekaran, P.; Stieber, S. C. E.; Collins, T. J.; Que, L.; Neese, F.; DeBeer, S. Prediction of high-valent iron K-edge absorption spectra by time-dependent Density Functional Theory. *Dalton Trans.*, **2011**, 40, 11070–11079.
4. Beckwith, M. A.; Roemelt, M.; Collomb, M. N.; Duboc, C.; Weng, T. C.; Bergmann, U.; Glatzel, P.; Neese, F.; DeBeer, S. Manganese K beta X-ray Emission Spectroscopy As a Probe of Metal-Ligand Interactions. *Inorg. Chem.*, **2011**, 50, 8397–8409.
5. Lee, N.; Petrenko, T.; Bergmann, U.; Neese, F.; DeBeer, S. Probing Valence Orbital Composition with Iron K beta X-ray Emission Spectroscopy. *J. Am. Chem. Soc.*, **2010**, 132, 9715–9727.
6. DeBeer-George, S.; Neese, F. Calibration of Scalar Relativistic Density Functional Theory for the Calculation of Sulfur K-Edge X-ray Absorption Spectra. *Inorg. Chem.*, **2010**, 49, 1849–1853.

7. DeBeer-George, S.; Petrenko, T.; Neese, F. Prediction of Iron- K-edge Absorption Spectra using Time-Dependent Density Functional Theory. *J. Phys. Chem. A*, **2008**, 112, 12936–12943.

8. DeBeer-George, S.; Petrenko, T.; Neese, F. Time-dependent density functional calculations of ligand K-edge X-ray absorption spectra. *Inorg. Chim. Acta*, **2008**, 361, 965–972.

B.4 Applications that make use of or include the following

1. Atanasov, Mihail; Ganyushin, Dmitry; Pantazis, Dimitrios A.; Sivalingam, Kantharuban; Neese, Frank. Detailed Ab Initio First-Principles Study of the Magnetic Anisotropy in a Family of Trigonal Pyramidal Iron(II) Pyrrolide Complexes. *Inorg. Chem.*, **2011**, 50 (16), 7460–7477. DOI: 10.1021/ic200196k.

2. Maganas, Dimitrios; Sottini, Silvia; Kyritsis, Panayotis; Groenen, Edgar J. J.; Neese, Frank. Theoretical Analysis of the Spin Hamiltonian Parameters in Co(II)S_4 Complexes, Using Density Functional Theory and Correlated ab initio Methods. *Inorg. Chem.*, **2011**, 50 (18), 8741–8754. DOI: 10.1021/ic200299y.

3. Petrenko, T.; Ray, K.; Wieghardt, K.; Neese, F. Vibrational Markers for the Open-Shell Character of Metal bis-Dithiolenes: An Infrared, resonance Raman and Quantum Chemical Study. *J. Am. Chem. Soc.*, **2006**, 128, 4422–4436.

4. Neese, F.; Pantazis, D. A. What is not required to make a single molecule magnet. *Faraday Discuss.*, **2011**, 148, 229–238.

5. Kruse, H.; Mladek, A.; Gkionis, K.; Hansen, A.; Grimme, S.; Sponer, J. Quantum Chemical Benchmark Study on 46 RNA Backbone Families Using a Dinucleotide Unit. *J. Chem. Theory Comput.*, **2015**, 11, 4972.

6. Qu, Z.-W.; Hansen, A.; Grimme, S. Co-C Bond Dissociation Energies in Cobalamin Derivatives and Dispersion Effects: Anomaly or Just Challenging? *J. Chem. Theory Comput.*, **2015**, 11, 1037.

7. Hansen, A.; Bannwarth, C.; Grimme, S.; Petrović, P.; Werlé, C.; Djukic, J.-P. The Thermochemistry of London Dispersion-Driven Transition Metal Reactions: Getting the 'Right Answer for the Right Reason'. *ChemistryOpen*, **2014**, 3, 177.

8. Krewald, V.; Neese, F.; Pantazis, D. A. On the magnetic and spectroscopic properties of high-valent Mn_3CaO_4 cubanes as structural units of natural and artificial water oxidizing catalysts. *J. Am. Chem. Soc.*, **2013**, 135, 5726–5739.

9. Kampa, M.; Pandelia, M.-E.; Lubitz, W.; van Gastel, M.; Neese, F. A Metal-Metal Bond in the Light-Induced State of [NiFe] Hydrogenases with Relevance to Hydrogen Evolution. *J. Am. Chem. Soc.*, **2013**, 135, 3915–3925.

10. Pandelia, M.-E.; Bykov, D.; Izsák, R.; Infossi, P.; Giudici-Orticoni, M.-T.; Bill, E.; Neese, F.; Lubitz, W. Electronic structure of the unique [4Fe-3S] cluster in O_2-tolerant hydrogenases characterized by Fe-57 Mossbauer and EPR spectroscopy. *Proc. Natl. Acad. Sci. USA*, **2013**, 110, 483–488.

11. Atanasov, M.; Surawatanawong, P.; Wieghardt, K.; Neese, F. A theoretical study of zero-field splitting in Fe(IV)S_6 (S = 1) and Fe(III)S_6 (S = 1/2) core complexes, $[Fe^{IV}(Et_2dtc)_3\text{-}(mnt)]^{(n-1)-}$ and $[Fe^{III}(Et_2dtc)_3\text{-}(mnt)]^{n-}$ (n=0, 1, 2, 3): The origin of the magnetic anisotropy. *Coord. Chem. Rev.*, **2013**, 257 (1), 27–41.

12. Retegan, M.; Collomb, M.-N.; Neese, F.; Duboc, C. A combined high-field EPR and quantum chemical study on a weakly ferromagnetically coupled dinuclear Mn(III) complex. A complete analysis of the EPR spectrum beyond the strong coupling limit. *Phys. Chem. Chem. Phys.*, **2013**, 15, 223–234.

13. Zadrozny, J. M.; Atanasov, M.; Bryan, A. M.; Lin, C. Y.; Rekken, B. D.; Power, P. P.; Neese, F.; Long, J. R. Slow magnetization dynamics in a series of two-coordinate iron(II) complexes. *Chem. Sci.*, **2013**, 4, 125–138.

14. Weber, K.; Krämer, T.; Shafaat, H. S.; Weyhermüller, T.; Bill, E.; van Gastel, M.; Neese, F.; Lubitz, W. A Functional [NiFe]-Hydrogenase Model Compound That Undergoes Biologically Relevant Reversible Thiolate Protonation. *J. Am. Chem. Soc.*, **2012**, 134, 20745–20755.

15. Kampa, M.; Lubitz, W.; van Gastel, M.; Neese, F. Computational study of the electronic structure and magnetic properties of the Ni-C state in [NiFe] hydrogenases including the second coordination sphere. *J. Biol. Inorg. Chem.*, **2012**, 17, 1269–1281.

16. Atanasov, M.; Comba, P.; Helmle, S.; Müller, D.; Neese, F. Zero-Field Splitting in a Series of Structurally Related Mononuclear Ni^{II}-Bispidine Complexes. *Inorg. Chem.*, **2012**, 51, 12324–12335.

17. Shafaat, H. S.; Weber, K.; Petrenko, T.; Neese, F.; Lubitz, W. Key Hydride Vibrational Modes in [NiFe] Hydrogenase Model Compounds Studied by Resonance Raman Spectroscopy and Density Functional Calculations. *Inorg. Chem.*, **2012**, 51, 11787–11797.

18. Argirevic, T.; Riplinger, C.; Stubbe, J.; Neese, F.; Bennati, M. ENDOR Spectroscopy and DFT Calculations: Evidence for the Hydrogen-Bond Network Within α2 in the PCET of E. coli Ribonucleotide Reductase. *J. Am. Chem. Soc.*, **2012**, 134, 17661–17670.

19. Albrecht, C.; Shi, L. L.; Perez, J. M.; van Gastel, M.; Schwieger, S.; Neese, F.; Streubel, R. Deoxygenation of Coordinated Oxaphosphiranes: A New Route to P=C Double-Bond Systems. *Chem. Eur. J.*, **2012**, 18, 9780–9783.

20. Maganas, D.; Krzystek, J.; Ferentinos, E.; Whyte, A. M.; Robertson, N.; Psycharis, V.; Terzis, A.; Neese, F.; Kyritsis, P. Investigating Magnetostructural Correlations in the Pseudooctahedral trans-$Ni^{II}(OPPh_2)(EPPh_2)N_2(sol)_2$ Complexes (E = S, Se; sol = DMF, THF) by Magnetometry, HFEPR, and ab Initio Quantum Chemistry. *Inorg. Chem.*, **2012**, 51, 7218–7231.

21. Ye, S. F.; Neese, F. How Do Heavier Halide Ligands Affect the Signs and Magnitudes of the Zero-Field Splittings in Halogenonickel(II) Scorpionate Complexes? A Theoretical Investigation Coupled to Ligand-Field Analysis. *J. Chem. Theory Comput.*, **2012**, 8, 2344–2351.

22. Nesterov, V.; Ozbolat-Schon, A.; Schnakenburg, G.; Shi, L. L.; Cangonul, A.; van Gastel, M.; Neese, F.; Streubel, R. An Unusal Case of Facile Non-Degenerate P-C Bond Making and Breaking. *Chem. Asian J.*, **2012**, 7, 1708–1712.

23. Bykov, D.; Neese, F. Reductive activation of the heme iron-nitrosyl intermediate in the reaction mechanism of cytochrome c nitrite reductase: a theoretical study. *J. Biol. Inorg. Chem.*, **2012**, 17, 741–760.

24. Lancaster, K. M.; Zaballa, M. E.; Sproules, S.; Sundararajan, M.; DeBeer, S.; Richards, J. H.; Vila, A. J.; Neese, F.; Gray, H. B. Outer-Sphere Contributions to the Electronic Structure of Type Zero Copper Proteins. *J. Am. Chem. Soc.*, **2012**, 134, 8241–8253.

25. Benkhauser-Schunk, C.; Wezisla, B.; Urbahn, K.; Kiehne, U.; Daniels, J.; Schnakenburg, G.; Neese, F.; Lutzen, A. Synthesis, Chiral Resolution, and Absolute Configuration of Functionalized Troger's Base Derivatives: Part II. *ChemPlusChem*, **2012**, 77, 396–403.

26. Ye, S. F.; Riplinger, C.; Hansen, A.; Krebs, C.; Bollinger, J. M.; Neese, F. Electronic Structure Analysis of the Oxygen-Activation Mechanism by Fe^{II}- and α-Ketoglutarate (αKG)-Dependent Dioxygenases. *Chem. Eur. J.*, **2012**, 18, 6555–6567.

27. Desrochers, P. J.; Sutton, C. A.; Abrams, M. L.; Ye, S. F.; Neese, F.; Telser, J.; Ozarowski, A.; Krzystek, J. Electronic Structure of Nickel(II) and Zinc(II) Borohydrides from Spectroscopic Measurements and Computational Modeling. *Inorg. Chem.*, **2012**, 51, 2793–2805.

28. Torres-Alacan, J.; Krahe, O.; Filippou, A. C.; Neese, F.; Schwarzer, D.; Vöhringer, P. The Photochemistry of $[Fe^{III}N_3(cyclam\text{-}ac)]PF_6$ at 266 nm. *Chem. Eur. J.*, **2012**, 18, 3043–3055.

29. Maekawa, M.; Römelt, M.; Daniliuc, C. G.; Jones, P. G.; White, P. S.; Neese, F.; Walter, M. D. Reactivity studies on $[Cp'MnX(thf)]_2$: manganese amide and polyhydride synthesis. *Chem. Sci.*, **2012**, 3, 2972–2979.

30. Pantazis, D. A.; Ames, W.; Cox, N.; Lubitz, W.; Neese, F. Two interconvertible structures that explain the spectroscopic properties of the oxygen-evolving complex of photosystem II in the S_2 state. *Angew. Chem. Int. Ed.*, **2012**, 51, 9935–9940. selected as cover article and VIP paper.

31. Vennekate, H.; Schwarzer, D.; Torres-Alacan, J.; Krahe, O.; Filippou, A. C.; Neese, F.; Vöhringer, P. Ultrafast primary processes of an iron-(III) azido complex in solution induced with 266 nm light. *Phys. Chem. Chem. Phys.*, **2012**, 14, 6165–6172.

32. Christian, G. J.; Ye, S.; Neese, F. Oxygen activation in extradiol catecholate dioxygenases – a density functional study. *Chem. Sci.*, **2012**, 3, 1600–1611.

33. Cowley, Ryan E.; Christian, Gemma J.; Brennessel, William W.; Neese, Frank; Holland, Patrick L. A Reduced (β-Diketiminato)iron Complex with End-On and Side-On Nitriles: Strong Backbonding or Ligand Non-Innocence? *Eur. J. Inorg. Chem.*, **2012**, 2012 (3), 479–483. DOI: 10.1002/ejic.201100787.

34. Thiessen, A.; Wettach, H.; Meerholz, K.; Neese, F.; Hoger, S.; Hertel, D. Control of electronic properties of triphenylene by substitution. *Organic Electronics*, **2012**, 13, 71–83.

35. Lancaster, K. M.; Roemelt, M.; Ettenhuber, P.; Hu, Y. L.; Ribbe, M. W.; Neese, F.; Bergmann, U.; DeBeer, S. X-ray Emission Spectroscopy Evidences a Central Carbon in the Nitrogenase Iron-Molybdenum Cofactor. *Science*, **2011**, 334, 974–977.

36. Ames, W.; Pantazis, D. A.; Krewald, V.; Cox, N.; Messinger, J.; Lubitz, W.; Neese, F. Theoretical Evaluation of Structural Models of the S_2 State in the Oxygen Evolving Complex of Photosystem II: Protonation States and Magnetic Interactions. *J. Am. Chem. Soc.*, **2011**, 133, 19743–19757.

37. Antony, J.; Grimme, S.; Liakos, D. G.; Neese, F. Protein-Ligand Interaction Energies with Dispersion Corrected Density Functional Theory and High-Level Wave Function Based Methods. *J. Phys. Chem. A*, **2011**, 115, 11210–11220.

38. Radoul, M.; Bykov, D.; Rinaldo, S.; Cutruzzola, F.; Neese, F.; Goldfarb, D. Dynamic Hydrogen-Bonding Network in the Distal Pocket of the Nitrosyl Complex of Pseudomonas aeruginosa cd(1) Nitrite Reductase. *J. Am. Chem. Soc.*, **2011**, 133, 3043–3055.

39. Liakos, D. G.; Neese, F. Interplay of Correlation and Relativistic Effects in Correlated Calculations on Transition-Metal Complexes: The $Cu_2O_2^{2+}$ Core Revisited. *J. Chem. Theory Comput.*, **2011**, 7, 1511–1523.

40. Riplinger, C.; Neese, F. The Reaction Mechanism of Cytochrome P450 NO Reductase: A Detailed Quantum Mechanics/Molecular Mechanics Study. *ChemPhysChem*, **2011**, 12, 3192–3203.

41. Surawatanawong, P.; Sproules, S.; Neese, F.; Wieghardt, K. Electronic Structures and Spectroscopy of the Electron Transfer Series $Fe(NO)L_2^z$ (z=1+, 0, 1-, 2-, 3-; L = Dithiolene). *Inorg. Chem.*, **2011**, 50, 12064–12074.

42. Cox, N.; Ames, W.; Epel, B.; Kulik, L. V.; Rapatskiy, L.; Neese, F.; Messinger, J.; Wieghardt, K.; Lubitz, W. Electronic Structure of a Weakly Antiferromagnetically Coupled Mn(II)Mn(III) Model Relevant to Manganese Proteins: A Combined EPR, ^{55}Mn-ENDOR, and DFT Study. *Inorg. Chem.*, **2011**, 50, 8238–8251.

43. Rota, J. B.; Knecht, S.; Fleig, T.; Ganyushin, D.; Saue, T.; Neese, F.; Bolvin, H. Zero field splitting of the chalcogen diatomics using relativistic correlated wave-function methods. *J. Chem. Phys.*, **2011**, 135, 114106.

44. Cox, N.; Rapatskiy, L.; Su, J. H.; Pantazis, D. A.; Sugiura, M.; Kulik, L.; Dorlet, P.; Rutherford, A. W.; Neese, F.; Boussac, A.; Lubitz, W.; Messinger, J. Effect of Ca^{2+}/Sr^{2+} Substitution on the Electronic Structure of the Oxygen-Evolving Complex of Photosystem II: A Combined Multifrequency EPR, ^{55}Mn-ENDOR, and DFT Study of the S_2 State. *J. Am. Chem. Soc.*, **2011**, 133, 3635–3648.

45. Maurice, R.; Sivalingam, K.; Ganyushin, D.; Guihery, N.; de Graaf, C.; Neese, F. Theoretical Determination of the Zero-Field Splitting in Copper Acetate Monohydrate. *Inorg. Chem.*, **2011**, 50, 6229–6236.

46. Su, J. H.; Cox, N.; Ames, W.; Pantazis, D. A.; Rapatskiy, L.; Lohmiller, T.; Kulik, L. V.; Dorlet, P.; Rutherford, A. W.; Neese, F.; Boussac, A.; Lubitz, W.; Messinger, J. The electronic structures of the S_2 states of the oxygen-evolving complexes of photosystem II in plants and cyanobacteria in the presence and absence of methanol. *Biochim. Biophys. Acta-Bioenergetics*, **2011**, 1807, 829–840.

47. Bykov, D.; Neese, F. Substrate binding and activation in the active site of cytochrome c nitrite reductase: a density functional study. *J. Biol. Inorg. Chem.*, **2011**, 16, 417–430.

48. Gennari, M.; Orio, M.; Pecaut, J.; Bothe, E.; Neese, F.; Collomb, M. N.; Duboc, C. Influence of Mixed Thiolate/Thioether versus Dithiolate Coordination on the Accessibility of the Uncommon +I and +III Oxidation States for the Nickel Ion: An Experimental and Computational Study. *Inorg. Chem.*, **2011**, 50, 3707–3716.

49. Ye, S. F.; Neese, F. Nonheme oxo-iron(IV) intermediates form an oxyl radical upon approaching the C-H bond activation transition state. *Proc. Natl. Acad. Sci. USA*, **2011**, 108, 1228–1233.

50. Gennari, M.; Pecaut, J.; DeBeer, S.; Neese, F.; Collomb, M. N.; Duboc, C. A Fully Delocalized Mixed-Valence Bis-μ-(Thiolato) Dicopper Complex: A Structural and Functional Model of the Biological Cu(A) Center. *Angew. Chem., Int. Ed.*, **2011**, 50, 5661–5665.

51. Gennari, M.; Retegan, M.; DeBeer, S.; Pecaut, J.; Neese, F.; Collomb, M. N.; Duboc, C. Experimental and Computational Investigation of Thiolate Alkylation in Ni(II) and Zn(II) Complexes: Role of the Metal on the Sulfur Nucleophilicity. *Inorg. Chem.*, **2011**, 50, 10047–10055.

52. Atanasov, M.; Delley, B.; Neese, F.; Tregenna-Piggott, P. L.; Sigrist, M. Theoretical Insights into the Magnetostructural Correlations in Mn(3)-Based Single-Molecule Magnets. *Inorg. Chem.*, **2011**, 50, 2112–2124.

53. Lassalle-Kaiser, B.; Hureau, C.; Pantazis, D. A.; Pushkar, Y.; Guillot, R.; Yachandra, V. K.; Yano, J.; Neese, F.; Anxolabéhère-Mallart, E. Activation of a water molecule using a mononuclear Mn complex: from Mn-aquo, to Mn-hydroxo, to Mn-oxyl via charge compensation. *Energy Environ. Sci.*, **2010**, 3, 924–938.

54. Pantazis, D. A.; Krewald, V.; Orio, M.; Neese, F. Theoretical magnetochemistry of dinuclear manganese complexes: broken symmetry density functional theory investigation on the influence of bridging motifs on structure and magnetism. *Dalton Trans.*, **2010**, 39, 4959–4967.

55. Woertink, J. S.; Tian, L.; Maiti, D.; Lucas, H. R.; Himes, R. A.; Karlin, K. D.; Neese, F.; Wurtele, C.; Holthausen, M. C.; Bill, E.; Sundermeyer, J.; Schindler, S.; Solomon, E. I. Spectroscopic and Computational Studies of an End-on Bound Superoxo-Cu(II) Complex: Geometric and Electronic Factors That Determine the Ground State. *Inorg. Chem.*, **2010**, 49, 9450–9459.

56. McNaughton, R. L.; Roemelt, M.; Chin, J. M.; Schrock, R. R.; Neese, F.; Hoffman, B. M. Experimental and Theoretical EPR Study of Jahn–Teller-Active HIPTN(3)N MoL Complexes (L = N_2, CO, NH_3). *J. Am. Chem. Soc.*, **2010**, 132, 8645–8656.

57. Geng, C. Y.; Ye, S. F.; Neese, F. Analysis of Reaction Channels for Alkane Hydroxylation by Nonheme Iron(IV)-Oxo Complexes. *Angew. Chem., Int. Ed.*, **2010**, 49, 5717–5720.

58. Orio, M.; Jarjayes, O.; Kanso, H.; Philouze, C.; Neese, F.; Thomas, F. X-Ray Structures of Copper(II) and Nickel(II) Radical Salen Complexes: The Preference of Galactose Oxidase for Copper(II). *Angew. Chem., Int. Ed.*, **2010**, 49, 4989–4992.

59. Gennari, M.; Orio, M.; Pecaut, J.; Neese, F.; Collomb, M. N.; Duboc, C. Reversible Apical Coordination of Imidazole between the Ni(III) and Ni(II) Oxidation States of a Dithiolate Complex: A Process Related to the Ni Superoxide Dismutase. *Inorg. Chem.*, **2010**, 49, 6399–6401.

60. Ye, S. F.; Price, J. C.; Barr, E. W.; Green, M. T.; Bollinger, J. M.; Krebs, C.; Neese, F. Cryoreduction of the NO-Adduct of Taurine:alpha-Ketoglutarate Dioxygenase (TauD) Yields an Elusive FeNO Species. *J. Am. Chem. Soc.*, **2010**, 132, 4739–4751.

61. Maganas, D.; Grigoropoulos, A.; Staniland, S. S.; Chatziefthimiou, S. D.; Harrison, A.; Robertson, N.; Kyritsis, P.; Neese, F. Tetrahedral and Square Planar $Ni(SPR_2)_2N_2$ complexes, R = Ph & iPr Revisited: Experimental and Theoretical Analysis of Interconversion Pathways, Structural Preferences, and Spin Delocalization. *Inorg. Chem.*, **2010**, 49, 5079–5093.

62. Anoop, A.; Thiel, W.; Neese, F. A Local Pair Natural Orbital Coupled Cluster Study of Rh Catalyzed Asymmetric Olefin Hydrogenation. *J. Chem. Theory Comput.*, **2010**, 6, 3137–3144.

63. Duboc, C.; Collomb, M.-N.; Pecaut, J.; Deronzier, A.; Neese, F. Understanding the Zero-Field Splitting of Mononuclear Manganese(II) Complexes from Combined EPR Spectroscopy and Quantum Chemistry. *Appl. Magn. Res.*, **2010**, 37, 229–245.

64. Ye, S. F.; Neese, F. The Unusual Electronic Structure of Dinitrosyl Iron Complexes. *J. Am. Chem. Soc.*, **2010**, 132, 3646–3647.

65. Kochem, A.; Orio, M.; Jarjayes, O.; Neese, F.; Thomas, F. Unsymmetrical one-electron oxidized Ni(II)-bis(salicylidene) complexes: a protonation-induced shift of the oxidation site. *Chem. Commun.*, **2010**, 46, 6765–6767.

66. Ozbolat-Schon, A.; Bode, M.; Schnakenburg, G.; Anoop, A.; van Gastel, M.; Neese, F.; Streubel, R. Insights into the Chemistry of Transient P-Chlorophosphanyl Complexes. *Angew. Chem., Int. Ed.*, **2010**, 49, 6894–6898.

67. Vancoillie, S.; Chalupsky, J.; Ryde, U.; Solomon, E. I.; Pierloot, K.; Neese, F.; Rulisek, L. Multireference Ab Initio Calculations of g tensors for Trinuclear Copper Clusters in Multicopper Oxidases. *J. Phys. Chem. B*, **2010**, 114, 7692–7702.

68. Grote, D.; Finke, C.; Kossmann, S.; Neese, F.; Sander, W. 3,4,5,6-Tetrafluorophenylnitren-2-yl: A Ground-State Quartet Triradical. *Chem. Eur. J.*, **2010**, 16, 4496–4506.

69. Ye, S. F.; Neese, F.; Ozarowski, A.; Smirnov, D.; Krzystek, J.; Telser, J.; Liao, J. H.; Hung, C. H.; Chu, W. C.; Tsai, Y. F.; Wang, R. C.; Chen, K. Y.; Hsu, H. F. Family of V(III)-Tristhiolato Complexes Relevant to

Functional Models of Vanadium Nitrogenase: Synthesis and Electronic Structure Investigations by Means of High-Frequency and -Field Electron Paramagnetic Resonance Coupled to Quantum Chemical Computations. *Inorg. Chem.*, **2010**, 49, 977–988.

70. Hegele, P.; Santhamma, B.; Schnakenburg, G.; Frohlich, R.; Kataeva, O.; Nieger, M.; Kotsis, K.; Neese, F.; Dotz, K. H. Hydroquinoid Chromium Complexes Bearing an Acyclic Conjugated Bridge: Chromium-Templated Synthesis, Molecular Structure, and Haptotropic Metal Migration. *Organometallics*, **2010**, 29, 6172–6185.

71. Ye, S. F.; Neese, F. Accurate Modeling of Spin-State Energetics in Spin-Crossover Systems with Modern Density Functional Theory. *Inorg. Chem.*, **2010**, 49, 772–774.

72. Orio, M.; Philouze, C.; Jarjayes, O.; Neese, F.; Thomas, F. Spin Interaction in Octahedral Zinc Complexes of Mono- and Diradical Schiff and Mannich Bases. *Inorg. Chem.*, **2010**, 49, 646–658.

73. Pantazis, D. A.; Orio, M.; Petrenko, T.; Zein, S.; Lubitz, W.; Messinger, J.; Neese, F. Structure of the Oxygen-Evolving Complex of Photosystem II: Information on the S_2 state through Quantum Chemical Calculation of its Magnetic Properties. *Phys. Chem. Chem. Phys.*, **2009**, 11, 6788–6798.

74. Baffert, C.; Orio, M.; Pantazis, D. A.; Duboc, C.; Blackman, A. G.; Blondin, G.; Neese, F.; Deronzier, A.; Collomb, M.-N. A trinuclear terpyridine frustrated spin system with a $Mn^{IV}{}_3O_4$ core: synthesis, physical characterization and quantum chemical modeling of its magnetic properties. *Inorg. Chem.*, **2009**, 48, 10281–10288.

75. Liakos, D.; Neese, F. A multiconfigurational ab initio study of the zero-field splitting in the di- and trivalent hexaquo-chromium complexes. *Inorg. Chem.*, **2009**, 48, 10572–10580.

76. Astashkin, A. V.; Klein, E. C.; Ganyushin, D.; Johnson.Winters, K.; Neese, F.; Kappler, U.; Enemark, J. H. Exchangeable oxygens in the vicinity of the molybdenum center of the high-pH form of sulfite oxidase and sulfite dehydrogenase. *Phys. Chem. Chem. Phys.*, **2009**, 11, 6733–6742.

77. Orio, M.; Pantazis, D. A.; Petrenko, T.; Neese, F. Magnetic and spectroscopic properties of mixed valence manganese(III,IV) dimers: a systematic study using broken symmetry density functional theory. *Inorg. Chem.*, **2009**, 48, 7251–7260.

78. Klein, E. L.; Astashkin, A. V.; Ganyushin, D.; Johnson-Winters, K.; Wilson, H. L.; Rajagopalan, K. V.; Neese, F.; Enemark, J. H. Direct Detection and Characterization of Chloride in the Active Site of the Low-pH Form of Sulfite Oxidase Using ESEEM Spectroscopy, Isotopic Labeling, and DFT Calculations. *Inorg. Chem.*, **2009**, 48 (11), 4743–4752.

79. Vancoillie, S.; Rulisek, L.; Neese, F.; Pierloot, K. Theoretical description of the structure and magnetic properties of nitroxide-Cu(II)-nitroxide spin triads. *J. Phys. Chem.*, **2009**, 113, 6149–6157.

80. Cowley, R. E.; Bill, E.; Neese, F.; Brennessel, W. W.; Holland, P. L. Iron(II) Complexes With Redox-Active Tetrazene (RNNNNR) Ligands. *Inorg. Chem.*, **2009**, 48, 4828–4836.

81. Gansäuer, A.; Fleckhaus, A.; Lafon, A.; Okkel, M.; Anakuthil, A.; Kotsis, K.; Neese, F. Catalysis via Homolytic Substitutions with C-O and Ti-O Bonds: Oxidative Additions and Reductive Eliminations in Single Electron Steps. *J. Am. Chem. Soc.*, **2009**, 131, 16989–16999.

82. Ye, S.; Neese, F. Quantum Chemical Studies of C-H Activation Reactions by High-Valent Nonheme Iron Centers. *Curr. Op. Chem. Biol.*, **2009**, 13 (1), 89–98.

83. Krahe, O.; Neese, F.; Streubel, R. The quest for ring-opening of oxaphosphirane complexes: a coupled cluster and density functional study of CH_3PO isomers and their $Cr(CO)_5$ complexes. *Chem. Eur. J.*, **2009**, 15, 2594–2601.

84. Romain, S.; Duboc, C.; Neese, F.; Riviere, E.; Hanton, L. R.; Blackman, A. G.; Philouze, C.; Lepretre, J. C.; Deronzier, A.; Collomb, M. N. An Unusual Stable Mononuclear Mn(III) Bis-terpyridine Complex Exhibiting Jahn-Teller Compression: Electrochemical Synthesis, Physical Characterisation and Theoretical Study. *Chem. Eur. J.*, **2009**, 15, 980–988.

85. Zein, S.; Neese, F. Ab initio and Coupled Perturbed DFT Calculation of Zero-Field Splittings in Mn(II) Transition Metal complexes. *J. Phys. Chem. A*, **2008**, 112, 7976–7983.

86. Ye, S.; Tuttle, T.; Bill, E.; Gross, Z.; Thiel, W.; Neese, F. The Noninnocence of Iron Corroles: A combined Experimental and Quantum Chemical Study. *Chem. Eur. J.*, **2008**, 34, 10839–10851. selected as very important paper.

87. Duboc, C.; Collomb, M.-N.; Pecaut, J.; Deronzier, A.; Neese, F. Definition of Magneto-Structural Correlations for the Mn(II) Ion. *Chem. Eur. J.*, **2008**, 21, 6498–6509.

88. Berry, J. F.; DeBeer-George, S.; Neese, F. Electronic Structure and Spectroscopy of "Superoxidized" Iron Centers in Model Systems: Theoretical and Experimental Trends. *Phys. Chem. Chem. Phys.*, **2008**, 10, 4361–4374.

89. Sander, W.; Grote, D.; Kossmann, S.; Neese, F. 2.3.5.6-Tetrafluorophenylnitren-4-yl: EPR Spectroscopic Characterization of a Quartet Ground State Nitreno Radical. *J. Am. Chem. Soc.*, **2008**, 130, 4396–4403.

90. Scheifele, Q.; Riplinger, C.; Neese, F.; Weihe, H.; Barra, A. L.; Jurany, F.; Podlesnyak, A.; Tregenna-Piggot, P. W. L. Spectroscopic and Theoretical Study of a Mononuclear Mn(III) Bioinorganic Complex Exhibiting a Compressed Jahn-Teller Octahedron. *Inorg. Chem.*, **2008**, 47, 439–447.

91. Zein, S.; Kulik, L. V.; Yano, J.; Kern, J.; Zouni, A.; Yachandra, V. K.; Lubitz, W.; Neese, F.; Messinger, J. Focussing the View on Nature's Water Splitting Catalyst. *Phil. Trans. Roy. Soc. London B*, **2008**, 363, 1167–1177.

92. Zein, S.; Duboc, C.; Lubitz, W.; Neese, F. Theoretical Characterization of zero-Field Splittings in Mn(II) Complexes. *Inorg. Chem.*, **2008**, 47, 134–142.

93. Parker, D. J.; Hammond, D.; Davies, E. S.; Garner, C. D.; Benisvy, L.; McMaster, J.; Wilson, C.; Neese, F.; Bothe, E.; Bittl, R.; Teutloff, C. A stable H-bonded ortho-Thioether Phenoxyl-Radical: A Chemical and Spectroscopic Analogue of •Tyr_{272} in apo-Galactose Oxidase. *J. Biol. Inorg. Chem.*, **2007**, 101, 1859–1864.

94. Chlopek, K.; Muresan, N.; Neese, F.; Wieghardt, K. Electronic Structures of Five-Coordinate Complexes of Iron Containing Zero, One, or Two π Radical Ligands: A Broken Symmetry Density Functional Theoretical Study. *Chem. Eur. J.*, **2007**, 13, 8391–8403.

95. Muresan, N.; Chlopek, K.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Bis(α-diimine)nickel Complexes: Molecular and Electronic Structure of Three Members of the Electron-Transfer Series $[Ni(L)_2]^z$ (z = 0, 1+, 2+) (L = 2-Phenyl-1,4-bis(isopropyl)-1,4-diazabutadiene). A Combined Experimental and Theoretical Study. *Inorg. Chem.*, **2007**, 46, 4905–4916.

96. Ray, K.; Petrenko, T.; Wieghardt, K.; Neese, F. Joint Spectroscopic and Theoretical Investigations of Transition Metal Complexes Involving Non-Innocent Ligands. *Dalton Trans.*, **2007**, 16, 1552–1566. DOI: 10.1039/B700096K.

97. Sinnecker, S.; Svensen, N.; Barr, E. W.; Ye, S.; Bollinger, J. M.; Neese, F.; Krebs, C. Spectroscopic and Theoretical Evaluation of the Structure of the High-Spin Fe(IV)-Oxo Intermediates in Taurine:α-Ketoglutarate Dioxygenase from Escherichia coli and its His99Ala Ligand Variant. *J. Am. Chem. Soc.*, **2007**, 129, 6168–6179.

98. Duboc, C.; Phoeung, T.; Zein, S.; Pécaut, J.; Collomb, M.-N.; Neese, F. Origin of the zero field splitting in mononuclear dihalide Mn(II) complexes: an investigation by multifrequency high-field EPR and density functional theory (DFT). *Inorg. Chem.*, **2007**, 46, 4905–4916.

99. DeBeer-George, S.; Petrenko, T.; Aliaga-Alcade, N.; Bill, E.; Mienert, B.; Sturhan, W.; Ming, Y.; Wieghardt, K.; Neese, F. Characterization of a Genuine Iron(V)Nitrido Species by Nuclear Resonant Vibrational Spectroscopy Coupled to Density Functional Calculations. *J. Am. Chem. Soc.*, **2007**, 129, 11053–11060.

100. Lehnert, N. M.; Cornelissen, U.; Neese, F.; Ono, T.; Noguchi, Y.; Okamoto, K.-I.; Fujisawa, K. Synthesis and Spectroscopic Characterization of Cu(II)-Nitrite Complexes with Hydrotris(pyrazolyl)borate and Related Ligands. *Inorg. Chem.*, **2007**, 46, 3916–3933.

101. Carmieli, R.; Larsen, T.; Reed, G. H.; Zein, S.; Neese, F.; Goldfarb, D. The Catalytic Mn^{2+} Sites in the Enolase-Inhibitor Complex - Crystallography, Single Crystal EPR and DFT calculations. *J. Am. Chem. Soc.*, **2007**, 129, 4240–4252.

102. Kokatam, S.; Ray, K.; Pap, J.; Bill, E.; Geiger, W. E.; LeSuer, R. J.; Rieger, P. H.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Molecular and Electronic Structure of Square Planar Gold Complexes Containing Two

1,2-di(4-tert-butylphenyl)ethylene-1,2-dithiolato Ligands: $[Au(L)_2]^{1+/0/1-/2-}$. A Combined Experimental and Computational Study. *Inorg. Chem.*, **2007**, 46, 1100–1111.

103. Ray, K.; DeBeer-George, S.; Solomon, E. I.; Wieghardt, K.; Neese, F. Description of the Ground State Covalencies of the Bis(dithiolato)Transition Metal Complexes Using X-ray Absorption Spectral and Time-Dependent-Density-Functional Studies. *Chem. Eur. Journal*, **2007**, 13 (10), 2753. selected for cover picture.

104. Chalupský, J.; Neese, F.; Solomon, E. I.; Ryde, U.; Rulíšek, L. Identification of intermediates in the reaction cycle of multicopper oxidases by quantum chemical calculations of spectroscopic parameters. *Inorg. Chem.*, **2006**, 45, 11051–11059.

105. Bart, S. C.; Chłopek, K.; Bill, E.; Bouwkamp, B. W.; Lobkovsky, E.; Neese, F.; Wieghardt, K.; Chirik, P. J. Electronic Structure of Bis(imino)pyridine Iron Dichloride, Monochloride and Neutral Ligand Complexes: A Combined Structural, Spectroscopic and Computational Study. *J. Am. Chem. Soc.*, **2006**, 128, 13901–13912.

106. Patra, A. K.; Bill, E.; Bothe, E.; Chlopek, K.; Neese, F.; Weyhermüller, T.; Stobie, K.; Ward, M. D.; McCleverty, J. A.; Wieghardt, K. The Electronic Structure of Mononuclear Bis(1,2-diaryl-1,2-ethylenedithiolate)iron Complexes Containing a Fifth Cyanide or Phosphite Ligand: A Combined Experimental and Computational Study. *Inorg. Chem.*, **2006**, 45, 7877–7890.

107. Berry, J. F.; Bill, E.; Bothe, E.; DeBeer-George, S.; Mienert, B.; Neese, F.; Wieghardt, K. An Octahedral Coordination Complex of Iron(VI) – One Step Ahead of Nature? *Science*, **2006**, 312, 1937–1941.

108. Chłopek, K.; Bothe, E.; Neese, F.; Weyhermüller, T.; Wieghardt, K. The Molecular and Electronic Structures of Tetrahedral Complexes of Nickel and Cobalt Containing N,N'-Disubstituted, Bulky o-Diiminobenzosemiquinonate(1-) π-Radical Ligands. *Inorg. Chem.*, **2006**, 45, 6298–6307.

109. Kababya, S.; Nelson, J.; Calle, C.; Neese, F.; Goldfarb, D. The electronic structure of bi-nuclear mixed valent copper azacryptates derived from integrated advanced EPR and DFT calculations. *J. Am. Chem. Soc.*, **2006**, 128, 2017–2029.

110. Berry, J. F.; Bill, E.; Neese, F.; Garcia-Serres, R.; Weyhermüller, T.; Wieghardt, K. Effect of N-Methylation of Macrocyclic Amine Ligands on the Spin State of Fe(III): A Tale of Two Fluoro Complexes. *Inorg. Chem.*, **2006**, 45, 2027–2037.

111. Kapre, R.; Ray, K.; Sylvestre, I.; Weyhermüller, T.; DeBeer-George, S.; Neese, F.; Wieghardt, K. The Molecular and Electronic Structure of Oxo-bis(benzene-1,2-dithiolato)chromate(V) Monoanions. A Combined Experimental and Density Functional Study. *Inorg. Chem.*, **2006**, 45, 3499–3509.

112. Zhu, W.; Marr, A. C.; Wang, Q.; Neese, F.; Spencer, J. E.; Blake, A. J.; Cooke, P. A.; Wilson, C.; Schröder, M. Modulation of the Electronic Structure and the Ni-Fe Distance in Heterobimetallic Models for the Active Site in [NiFe]Hydrogenase: Is there a Ni-Fe Bond? *Proc. Natl. Acad. Sci. (USA)*, **2005**, 102, 18280–18285.

113. Astashkin, A. V.; Neese, F.; Raitsimaring, A. M.; Cooney, J. J. A.; Bultman, E.; Enemark, J. H. Pulsed EPR investigation of systems modelling molybdenum enzymes: hyperfine and quadrupole parameters of oxo-^{17}O in $[Mo^{17}O(SPh)_4]^-$. *J. Am. Chem. Soc.*, **2005**, 127, 16713–16722.

114. Benisvy, L.; Bittl, R.; Bothe, E.; Garner, C. D.; McMaster, J.; Ross, S.; Teutloff, C.; Neese, F. Phenoxyl Radicals Hydrogen-Bonded to Imidazolium – Analogues of Tyrosyl D• of Photosystem II: High-Field EPR and DFT Studies. *Angew. Chem. Int. Ed.*, **2005**, 44, 5314–5317.

115. Praneeth, V. K. K.; Neese, F.; Lehnert, N. Spin Density Distribution in Five- and Six-Coordinate Iron(II)-Porphyrin NO Complexes Evidenced by Magnetic Circular Dichroism Spectroscopy. *Inorg. Chem.*, **2005**, 44, 2570–2572.

116. Sinnecker, S.; Neese, F.; Lubitz, W. Dimanganese Catalase – Spectroscopic Parameters from Broken Symmetry Density Functional Theory of the Superoxidized Mn^{III}/Mn^{IV} state. *J. Biol. Inorg. Chem.*, **2005**, 10, 231–238.

117. Blanchard, S.; Neese, F.; Bothe, E.; Bill, E.; Weyhermüller, T.; Wieghardt, K. Square Planar vs. Tetrahedral Coordination in Diamagnetic Complexes of Nickel(II) Containing Two Bidentate π Radical Monoanions. *Inorg. Chem.*, **2005**, 44, 3636–3656.

118. Mader-Cosper, M.; Neese, F.; Astashkin, A. V.; Carducci, M. A.; Raitsimring, A. M.; Enemark, J. H. Determination of the Magnitude and Orientation of the g-Tensors for cis,trans-$(L\text{-}N_2S_2)Mo^VOX$ (X = Cl, SCH_2Ph) by Single Crystal EPR and Molecular Orbital Calculations. *Inorg. Chem.*, **2005**, 44, 1290–1301.

119. Fouqueau, A.; Casida, M. E.; Lawson, L. M.; Hauser, A.; Neese, F. Comparison of Density Functionals for Energy and Structural Differences Between the High-^{5}T2g: (t2g^4)(eg^2) and Low-^{1}A1g: (t2g^6)(eg^0) Spin States of Iron(II) Coordination Compounds: II. Comparison of Results for More than Ten Modern Functionals with Ligand Field Theory and Ab Initio Results for Hexaquoferrous Dication, $[Fe(H_2O)_6]^{2+}$ and Hexaminoferrous Dication $[Fe(NH_3)_6]^{2+}$. *J. Chem. Phys.*, **2005**, 122, 044110.

120. Aliaga-Alcade, N.; DeBeer George, S.; Bill, E.; Wieghardt, K.; Neese, F. The Geometric and Electronic Structure of [(Cyclam-acetato)Fe(N)]$^+$: a Genuine Iron(V) Species with Ground State Spin S = 1/2. *Angew. Chem. Int. Ed.*, **2005**, 44, 2908–2912.

121. Bill, E.; Bothe, E.; Chaudhuri, P.; Chlopek, K.; Herebian, D.; Kokatam, S.; Ray, K.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Molecular and Electronic Structure of Four- and Five-Coordinate Cobalt Complexes Containing Two o-Phenylendiamine- or Two o-Aminophenol-Type Ligands at Various Oxidation Levels: An Experimental, Density Functional and Correlated ab initio Study. *Chem. Eur. J.*, **2004**, 11, 204–224.

122. Paine, T.; Bothe, W.; Bill, E.; Weyhermüller, T.; Slep, L.; Neese, F.; Chaudhuri, P. Nonoxo Vanadium(IV) and Vanadyl(V) Complexes with Mixed O,X,O-Donor Ligand (X = S, Se, P, PO). *Inorg. Chem.*, **2004**, 43, 7324–7338.

123. Baute, D.; Arieli, D.; Zimmermann, H.; Neese, F.; Weckhuysen, B.; Goldfarb, D. The Structure of Copper Histidine Complexes in Solution and in Zeolite Y: A Combined X- and W-Band Pulsed EPR/ENDOR and DFT Study. *J. Am. Chem. Soc.*, **2004**, 126, 11733–11745.

124. Garcia Serres, R.; Grapperhaus, C. A.; Bothe, E.; Bill, E.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Structural, Spectroscopic and Computational Study of an Octahedral, Non-heme FeNO$^{6\,7\,8}$ Series: [Fe(NO)(cyclam-ac)]$^{2+/1+/0}$. *J. Am. Chem. Soc.*, **2004**, 126, 5138–5153.

125. Sinnecker, S.; Noodleman, L.; Neese, F.; Lubitz, W. Calculation of the EPR Parameters of a Mixed Valence Mn(III)/Mn(IV) Model Complex with Broken Symmetry Density Functional Theory. *J. Am. Chem. Soc.*, **2004**, 126, 2613–2622.

126. Sinnecker, S.; Neese, F.; Lubitz, W. Benzosemichinone Solvent Interactions. A Density Functional Study of Electric and Magnetic Properties for Probing Hydrogen Bond Strengths and Geometries. *J. Am. Chem. Soc.*, **2004**, 126, 3280–3290.

127. van Gastel, M.; Fichtner, C.; Neese, F.; Lubitz, W. EPR Experiments to Elucidate the Structure of the Ready and Unready States of the [NiFe] Hydrogenase of Desulfovibrio vulgaris Miyazaki F. *Biochem. Soc. Trans.*, **2005**, 33, 7–11.

128. van Gastel, M.; Lassman, G.; Lubitz, W.; Neese, F. The unusual EPR parameters of the cysteine radical: a DFT and correlated ab initio study. *J. Am. Chem. Soc.*, **2004**, 126, 2237–2246.

129. Fouqueau, A.; Mer, S.; Casida, M. E.; Daku, L. M. L.; Hauser, A.; Mieva, T.; Neese, F. Comparison of Density Functionals for Energy and Structural Differences between the High-^{5}T2g: t2g^{4}eg^2 and Low-^{1}A1g: t2g^{6}eg^0 Spin States of the Hexaquo-Ferrous Ion, $[Fe(H^2O)^6]^{2+}$. *J. Chem. Phys.*, **2004**, 120, 9473–9486.

130. Slep, L. D.; Mijovilovich, A.; Meyer-Klaucke, W.; Weyhermüller, T.; Bill, E.; Bothe, E.; Neese, F.; Wieghardt, K. The Mixed-valent Fe^{Iv}(μ-O)(μ-carboxylato)$_2Fe^{III3+}$ Core. *J. Am. Chem. Soc.*, **2003**, 125, 15554–15570.

131. Herebian, D.; Wieghardt, K.; Neese, F. Analysis and Interpretation of Metal-Radical Coupling in a Series of Square Planar Nickel Complexes. Correlated Ab Initio and Density Functional Investigation of $[Ni(L(ISQ))_2]$ (L(ISQ)=3,5-di-tert-butyl-odiiminobenzosemquinone). *J. Am. Chem. Soc.*, **2003**, 125, 10997–11005.

132. Herebian, D.; Bothe, E.; Neese, F.; Weyhermüller, T.; Wieghardt, K. The Molecular and Electronic Structures of Bis(o-diiminobenzosemiquinonato)metal(II) Complexes (Ni, Pd, Pt), their Monocations and Anions, and their Dimeric Dications Containing Weak Metal-Metal Bonds. *J. Am. Chem. Soc.*, **2003**, 125, 9116–9128.

133. Ghosh, P.; Bill, E.; Weyhermüller, T.; Neese, F.; Wieghardt, K. The non-Innocence of the Ligand Glyoxal-bis (2-mercaptoanil). The Electronic Structures of $[Fe(gma)]_2$, [Fe(gma)(py)]•py, $[Fe(gma)(CN)]^{1-/0}$, [Fe(gma)I], $[Fe(gma)(PR_3)$?] (n=1,2). Experimental and Theoretical Evidence for 'Excited State' Coordination. *J. Am. Chem. Soc.*, **2003**, 125, 1293–1308.

134. Einsle, O.; Messerschmidt, A.; Huber, R.; Kroneck, P. M. H.; Neese, F. Mechanism of the Six Electron Reduction of Nitrite to Ammonia by Cytochrome c Nitrite Reductase (CCNIR). *J. Am. Chem. Soc.*, **2002**, 124, 11737–11745.

135. Sun, X.; Chun, H.; Hildenbrand, K.; Bothe, E.; Weyhermüller, T.; Neese, F.; Wieghardt, K. o-Iminobenzosemiquinonato(1-) and o-Amidophenolato(2-) Complexes of Palladium(II) and Plantinum(II): A Combined Experimental and Density Functional Theoretical Study. *Inorg. Chem.*, **2002**, 41, 4295–4303.

136. Li, M.; Bonnet, D.; Bill, E.; Neese, F.; Weyhermüller, T.; Blum, N.; Sellmann, D.; Wieghardt, K. Tuning the Electronic Structure of Octahedral Iron Complexes [FeL(X)] (L = 1-alkyl-4,7-bis(4-tert-butyl-2-mercaptobenzyl)-1,4,7-triazacyclo-nonane, X = Cl, CH_3O, CN, CO). The S=1/2 to S=3/2 Spin-Equilibrium of [FeL(Pr)(NO)]. *Inorg. Chem.*, **2002**, 41, 3444–3456.

137. Lehnert, N.; Neese, F.; Ho, R. Y. N.; Que Jr., L.; Solomon, E. I. Electronic Structure and Reactivity of Low-Spin Fe(III)-Hydroperoxo Complexes: Comparison to Activated Bleomycin. *J. Am. Chem. Soc.*, **2002**, 124, 10810–10822.

138. Grapperhaus, C. A.; Bill, E.; Weyhermüller, T.; Neese, F.; Wieghardt, K. Electronic and Geometric Structure and Spectroscopy of a High Valent Manganese(V) Nitrido Complex. An Experimental and DFT Study. *Inorg. Chem.*, **2001**, 41, 4191–4198.

139. Neese, F.; Solomon, E. I. Detailed Spectroscopic and Theoretical Studies on $[Fe(EDTA)(O_2)]^{3-}$: the Electronic Structure of the Side-On Ferric Peroxide Bond and its Relevance to Reactivity. *J. Am. Chem. Soc.*, **1998**, 120, 12829–12848.

B.5 Reviews of interest

1. Atanasov, M.; Aravena, D.; Suturina, E.; Bill, E.; Maganas, D.; Neese, F. First principles approach to the electronic structure, magnetic anisotropy and spin relaxation in mononuclear 3d-transition metal single molecule magnets. *Coord. Chem. Rev.*, **2015**, 289, 177–214.

2. Bursch, Markus; Mewes, Jan-Michael; Hansen, Andreas; Grimme, Stefan. Best-Practice DFT Protocols for Basic Molecular Computational Chemistry. *Angew. Chem. Int. Ed.*, **2022**, 61 (42), e202205735. DOI: 10.1002/anie.202205735.

3. Bannwarth, Christoph; Caldeweyher, Eike; Ehlert, Sebastian; Hansen, Andreas; Pracht, Philipp; Seibert, Jakob; Spicher, Sebastian; Grimme, Stefan. Extended tight-binding quantum chemistry methods. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2021**, 11 (2), e1493. DOI: 10.1002/wcms.1493.

4. Ray, K.; Petrenko, T.; Wieghardt, K.; Neese, F. Joint Spectroscopic and Theoretical Investigations of Transition Metal Complexes Involving Non-Innocent Ligands. *Dalton Trans.*, **2007**, 16, 1552–1566. DOI: 10.1039/B700096K.

5. Neese, F.; Liakos, D. G.; Ye, S. F. Correlated Wavefunction Methods in Bioinorganic Chemistry. *J. Biol. Inorg. Chem.*, **2011**, 16, 821–829.

6. Neese, F.; Ames, W.; Christian, G.; Kampa, M.; Liakos, D. G.; Pantazis, D. A.; Roemelt, M.; Surawatanawong, P.; Ye, S. F. Dealing with Complexity in Open-Shell Transition Metal Chemistry from a Theoretical Perspective: Reaction Pathways, Bonding, Spectroscopy, and Magnetic Properties. *Adv. Inorg. Chem.*, **2010**, 62, 301–349.

7. Orio, M.; Pantazis, D. A.; Neese, F. Density Functional Theory. *Photosynth. Res.*, **2009**, 102, 443–453.

8. Neese, F. Density Functional Theory and EPR Spectroscopy: a guided tour. *EPR Newsletter*, **2009**, 18 (4), Pro & Contra section. Pro & Contra section.

9. Neese, F. Prediction of Molecular Spectra and Molecular Properties with Density Functional Theory: from Fundamental Theory to Exchange Coupling. *Coord. Chem. Rev.*, **2009**, 253, 526–563.

10. Neese, Frank. *Spin-Hamiltonian Parameters from First Principle Calculations: Theory and Application*, pages 175–229. Springer New York, New York, NY, **2009**. DOI: 10.1007/978-0-387-84856-3_5.

11. Kirchner, B.; Wennmohs, F.; Ye, S.; Neese, F. Theoretical Bioinorganic Chemistry: Electronic Structure Makes a Difference. *Curr. Op. Chem. Biol.*, **2007**, 11, 131–141.

12. Neese, F.; Petrenko, T.; Ganyushin, D.; Olbrich, G. Advanced Aspects of ab initio Theoretical Spectroscopy of Open-Shell Transition Metal Ions. *Coord. Chem. Rev.*, **2007**, 205, 288–327.

13. Ye, S.; Neese, F. Combined Quantum Chemical and Spectroscopic Studies on Transition Metal Complexes with Coordinating Radicals. *Chemtracts (Special Volume on Computational Inorganic Chemistry)*, **2006**, 19, 77–86.

14. Sinnecker, S.; Neese, F. Theoretical Bioinorganic Spectroscopy. In Reiher, M., editor, *Current Topics in Chemistry*. Springer, Heidelberg, **2006**.

15. Neese, F. Quantum Chemical Approaches to Spin-Hamiltonian Parameters. Specialist Periodical Reports on EPR Spectroscopy Vol. 20, (Ed. B. Gilbert) Royal Society Press, **2006**.

16. Neese, F. A Critical Evaluation of DFT, including Time-Dependent DFT, Applied to Bioinorganic Chemistry. *J. Biol. Inorg. Chem.*, **2006**, 11, 702–711. commentary on invitation.

17. Neese, F.; Munzarova, M. L. Historical Aspects of EPR Parameter Calculations. In Kaupp, M.; Bühl, M.; Malkin, V., editors, *Calculation of NMR and EPR Parameters. Theory and Applications*, pages 21–32. Wiley-VCH, **2004**.

18. Neese, F. Zero-Field Splitting. In Kaupp, M.; Bühl, M.; Malkin, V., editors, *Calculation of NMR and EPR Parameters. Theory and Applications*, pages 541–564. Wiley-VCH, **2004**.

19. Neese, F. Application of EPR Parameter Calculations in Bioinorganic Chemistry. In Kaupp, M.; Bühl, M.; Malkin, V., editors, *Calculation of NMR and EPR Parameters. Theory and Applications*, pages 581–591. Wiley-VCH, **2004**.

20. Neese, F. Quantum Chemical Calculations of Spectroscopic Properties of Metalloproteins and Model Compounds: EPR and Mössbauer Properties. *Curr. Op. Chem. Biol.*, **2003**, 7, 125–135.

21. Neese, F.; Solomon, E. I. Calculation and Interpretation of Spin-Hamiltonian Parameters in Transition Metal Complexes. In Miller, J. S.; Drillon, M., editors, *Magnetoscience – From Molecules to Materials*, volume IV, pages 345–466. Wiley, **2003**.

APPENDIX

C

GLOSSARY

A

ADEX
Atomic Decomposition of Exchange

ADFT
Analytic Density Functional Theory

ADLD
Atomic Decomposition of London Dispersion

ALPB
Analytical Linearized Poisson-Boltzmann

B

BB
Boys-Bernardi

BSSE
Basis Set Superposition Error

BUPO
Bubblepoles

C

CC
Coupled-Cluster

cc-pVXZ
Dunning's Corrleation Consistent Polarized Valence Basis Set Family of x-tuple Zeta Quality

CCSD
Coupled-Cluster with Singles and Doubles

CCSD(T)
Coupled-Cluster with Singles, Doubles and Perturbative Triples

CI
Configuration Interaction

CNDO
Complete Neglect of Differential Overlap

CP-SCF
Coupled Perturbed Self Consisten Field

CPCM
Conductor-like Continuum Polarization Model

CSF
Configuration State Function

D

DFT
Density Functional Theory

DIIS
Direct Inversion in Iterative Subspace

DLPNO
Domain-Based Local Pair Natural Orbital

DMRG
Density Matrix Renormalization Group

E

ECP
Effective Core Potential

EPR
Electron Paramagnetic Resonance

F

FB
Foster Boys Orbital Localization

FCI
Full Configuration Interaction

G

gCP
Geometrical Counterpoise

GGA
Generalized Gradient Approximation

GS-ROCIS
General-spin Restricted Open-Shell Configuration Interaction Singles

GVPT2
Generalized Vibrational Perturbation Theory of Second Order

H

HF
Hartree-Fock

HOMO
Highest Occupied Molecular Orbital

I

ICE
Iterative Configuration Expansion

INDO
Intermediate Neglect of Differential Overlap

K

KDIIS
Kolmar's DIIS

L

LED
Local Energy Decomposition

LMO
Localized Molecular Orbital

LUMO
Lowest Unoccupied Molecular Orbital

M

MNDO
Modified Neglect of Diatomic Overlap

MO
Molecular Orbital

MP2
M\oller-Plesset Perturbation Theory of Second Order

N

NBO
Natural Bond Orbitals

NCS
Natural Chemical Shielding Analysis

NDDO
Neglect of Diatomic Differential Overlap

NDO
Neglect of Differential Overlap

NMR
Nuclear Magnetic Resonance

O

OO
Orbital-optimized

OPI
ORCA Python Interface

P

PCM
Polarizable Continuum Model

PM
Pipek Mezey Orbital Localization

PNO
Pair Natural Orbital

PP
Pseudopotential

Q

QRO
Quasi-restricted Orbitals

R

RHF
Restricted Hartree Fock

RI
Resolution of the Identity

RIXS
Resonant Inelastic X-ray Scattering

ROCIS
Restricted Open-Shell Configuration Interaction Singles

ROHF
Restricted Open-Shell Hartree Fock

RPA
Random Phase Approximation

S

SCF
Self-consistent Field

SOC
Spin-orbit Coupling

SSC
Spin-spin Coupling

T

TDA
Tamm-Dancoff Approximation

TRAH
Trust-Region Augmented Hessian

U

UCO
Unrestricted Corresponding Orbital

UHF
Unrestricted Hartree-Fock

UKS
Unrestricted Kohn-Sham

UNO
Unrestricted Natural Orbital

V

VCD
Vibrational Circular Dichroism

VPT2
Second-order Vibrational Perturbation Theory

W

WBO
Wiberg Bond Order

WFT
Wave Function Theory

X

X2C
Exact Two-Component

XAS
X-ray Absorption Spectroscopy

XES
X-ray Emission Spectroscopy

XMCD
X-ray Magnetic Circular Dichroism

XPS
X-ray Photoelectron Spectroscopy

Z

ZORA
Zeroth-Order Regular Approximation

ZPE
Zero-Point Energy

BIBLIOGRAPHY

[1] Zerner, M. C.; Hehenberger, M. *Chem. Phys. Lett.*, **1979**, 62, 550.

[2] Guest, M. F.; Saunders, V. R. *Mol. Phys.*, **1974**, 28, 819.

[3] Saunders, V. R.; Hillier, I. H. *Int. J. Quant. Chem.*, **1973**, VII, 699.

[4] Pulay, P. *Chem. Phys. Lett.*, **1980**, 73, 393.

[5] Pulay, P. Improved SCF Convergence Acceleration. *J. Comput. Chem.*, **0024**, 3 (4), 556–560. DOI: 10.1002/jcc.540030413.

[6] Kollmar, Christian. The role of energy denominators in self-consistent field (SCF) calculations for open shell systems. *J. Chem. Phys.*, **1996**, 105 (18), 8204–8212. DOI: 10.1063/1.472674.

[7] Kollmar, Christian. Convergence optimization of restricted open-shell self-consistent field calculations. *Int. J. Quant. Chem.*, **1997**, 62 (6), 617–637. DOI: 10.1002/(SICI)1097-461X(1997)62:6<617::AID-QUA2>3.0.CO;2-K.

[8] Fischer, T. H.; Almlöf, J. *J. Phys. Chem.*, **1992**, 96, 9768.

[9] Neese, F. Approximate Second Order Convergence for Spin Unrestricted Wavefunctions. *Chem. Phys. Lett.*, **2000**, 325, 93–98.

[10] Bacskay, George B. A Quadratically Convergent Hartree–Fock (QC-SCF) Method. Application to Closed Shell Systems. *Chem. Phys.*, **1981**, 61 (3), 385–404. DOI: 10.1016/0301-0104(81)85156-7.

[11] Sałek, Paweł; Høst, Stinne; Thøgersen, Lea; Jørgensen, Poul; Manninen, Pekka; Olsen, Jeppe; Jansík, Branislav; Reine, Simen; Pawłowski, Filip; Tellgren, Erik; Helgaker, Trygve; Coriani, Sonia. Linear-Scaling Implementation of Molecular Electronic Self-Consistent Field Theory. *J. Chem. Phys.*, **2007**, 126 (11), 114110.

[12] Høyvik, Ida-Marie; Jansik, Branislav; Jørgensen, Poul. Trust Region Minimization of Orbital Localization Functions. *J. Chem. Theory Comput.*, **2012**, 8 (9), 3137–3146. DOI: 10.1021/ct300473g.

[13] Helmich-Paris, Benjamin. A Trust-Region Augmented Hessian Implementation for Restricted and Unrestricted Hartree–Fock and Kohn–Sham Methods. *J. Chem. Phys.*, **2021**, 154 (16), 164104. DOI: 10.1063/5.0040798.

[14] Harding, D. J.; Gruene, P.; Haertelt, M.; Meijer, G.; Fielicke, A.; Hamilton, S. M.; Hopkins, W. S.; Mackenzie, S. R.; Neville, S. P.; Walsh, T. R. Probing the Structures of Gas-Phase Rhodium Cluster Cations by Far-Infrared Spectroscopy. *J. Chem. Phys.*, **2010**, 133 (21), 214304. DOI: 10.1063/1.3509778.

[15] Assfeld, X.; Rivail, J.-L. Quantum Chemical Computations on Parts of Large Molecules: The Ab Initio Local Self Consistent Field Method. *Chem. Phys. Lett.*, **1996**, 263, 100–106. DOI: 10.1016/S0009-2614(96)01165-7.

[16] Pritchard, Benjamin P.; Altarawy, Doaa; Didier, Brett; Gibson, Tara D.; Windus, Theresa L. New Basis Set Exchange: An Open, Up-to-Date Resource for the Molecular Sciences Community. *J. Chem. Inf. Model.*, **2019**, 59 (11), 4814–4820. DOI: 10.1021/acs.jcim.9b00725.

[17] Zheng, Jingjing; Xu, Xuefei; Truhlar, Donald G. *Theor. Chem. Acc.*, **2010**, 128, 295–305.

[18] Rappoport, Dmitrij; Furche, Filipp. *J. Chem. Phys.*, **2010**, 133, 134105.

[19] Rappoport, Dmitrij. Property-optimized Gaussian basis sets for lanthanides. *J. Chem. Phys.*, **2021**, 155, 124102. DOI: 10.1063/5.0065611.

[20] Weigend, Florian; Baldes, Alexander. Segmented Contracted Basis Sets for One- and Two-Component Dirac–Fock Effective Core Potentials. *J. Chem. Phys.*, **2010**, 133 (17), 174102. DOI: 10.1063/1.3495681.

[21] Jensen, Frank. Polarization Consistent Basis Sets: Principles. *J. Chem. Phys.*, **2001**, 115 (20), 9113–9125. DOI: 10.1063/1.1413524.

[22] Jensen, Frank. Polarization Consistent Basis Sets. II. Estimating the Kohn–Sham Basis Set Limit. *J. Chem. Phys.*, **2002**, 116 (16), 7372–7379. DOI: 10.1063/1.1465405.

[23] Jensen, Frank; Helgaker, Trygve. Polarization Consistent Basis Sets. V. The Elements Si–Cl. *J. Chem. Phys.*, **2004**, 121 (8), 3463–3470. DOI: 10.1063/1.1756866.

[24] Jensen, Frank. Polarization Consistent Basis Sets. 4: The Elements He, Li, Be, B, Ne, Na, Mg, Al, and Ar. *J. Phys. Chem. A*, **2007**, 111 (44), 11198–11204. DOI: 10.1021/jp068677h.

[25] Jensen, Frank. Polarization Consistent Basis Sets. VII. The Elements K, Ca, Ga, Ge, As, Se, Br, and Kr. *J. Chem. Phys.*, **2012**, 136 (11), 114107. DOI: 10.1063/1.3690460.

[26] Jensen, Frank. Unifying General and Segmented Contracted Basis Sets: Segmented Polarization Consistent Basis Sets. *J. Chem. Theory Comput.*, **2014**, 10 (3), 1074–1085. DOI: 10.1021/ct401026a.

[27] Jensen, Frank. Segmented Contracted Basis Sets Optimized for Nuclear Magnetic Shielding. *J. Chem. Theory Comput.*, **2015**, 11 (1), 132–138. DOI: 10.1021/ct5009526.

[28] Lehtola, Susi. Polarized Gaussian Basis Sets from One-Electron Ions. *J. Chem. Phys.*, **2020**, 152 (13), 134108. DOI: 10.1063/1.5144964.

[29] Yamamoto, Hironori; Matsuoka, Osamu. Accurately Energy-Optimized Gaussian Basis Sets for Hydrogen 1s through 5g Orbitals. *Bulletin of the University of Electro-Communications*, **1992**, 5, 23–34. In Japanese. Citation at https://ci.nii.ac.jp/naid/40004737908/en/.

[30] Noro, Takeshi; Sekiya, Masahiro; Koga, Toshikatsu. Correlating Basis Sets for the H Atom and the Alkali-Metal Atoms from Li to Rb. *Theor. Chem. Acc.*, **2003**, 109, 85–90. DOI: 10.1007/s00214-002-0425-z.

[31] Noro, Takeshi; Sekiya, Masahiro; Koga, Toshikatsu. Segmented Contracted Basis Sets for Atoms H through Xe: Sapporo-(DK)-nZP Sets (n = D, T, Q). *Theor. Chem. Acc.*, **2012**, 131, 1124. DOI: 10.1007/s00214-012-1124-z.

[32] Partridge, Harry. Near Hartree–Fock Quality GTO Basis Sets for the Second-Row Atoms. *J. Chem. Phys.*, **1987**, 87 (11), 6643–6647. DOI: 10.1063/1.453450.

[33] Partridge, Harry. Near Hartree–Fock Quality GTO Basis Sets for the First- and Third-Row Atoms. *J. Chem. Phys.*, **1989**, 90 (2), 1043–1047. DOI: 10.1063/1.456157.

[34] Pacios, Luis Fernandez; Christiansen, Peter A. Ab initio relativistic effective potentials with spin-orbit operators. I. Li through Ar. *J. Chem. Phys.*, **1985**, 82 (6), 2664–2671. DOI: 10.1063/1.448263.

[35] Hurley, M. M.; Pacios, Luis Fernandez; Christiansen, P. A.; Ross, R. B.; Ermler, W. C. Ab initio relativistic effective potentials with spin-orbit operators. II. K through Kr. *J. Chem. Phys.*, **1986**, 84 (12), 6840–6853. DOI: 10.1063/1.450689.

[36] LaJohn, L. A.; Christiansen, P. A.; Ross, R. B.; Atashroo, T.; Ermler, W. C. Ab initio relativistic effective potentials with spin-orbit operators. III. Rb through Xe. *J. Chem. Phys.*, **1987**, 87 (5), 2812–2824. DOI: 10.1063/1.453069.

[37] Ross, R. B.; Powers, J. M.; Atashroo, T.; Ermler, W. C.; LaJohn, L. A.; Christiansen, P. A. Ab initio relativistic effective potentials with spin-orbit operators. IV. Cs through Rn. *J. Chem. Phys.*, **1990**, 93 (9), 6654–6670. DOI: 10.1063/1.458934.

[38] Ross, R. B.; Gayen, Sanjukta; Ermler, W. C. Ab initio relativistic effective potentials with spin-orbit operators. V. Ce through Lu. *J. Chem. Phys.*, **1994**, 100 (11), 8145–8155. DOI: 10.1063/1.466809.

[39] Ermler, W. C.; Ross, R. B.; Christiansen, P. A. Ab initio relativistic effective potentials with spin-orbit operators. VI. Fr through Pu. *Int. J. Quantum Chem.*, **1991**, 40 (6), 829–846. DOI: 10.1002/qua.560400611.

[40] Nash, Clinton S.; Bursten, Bruce E.; Ermler, Walter C. Ab initio relativistic effective potentials with spin-orbit operators. VII. Am through element 118. *J. Chem. Phys.*, **1997**, 106 (12), 5133–5142. DOI: 10.1063/1.473992.

[41] Hay, P. Jeffrey; Wadt, W. R. Ab initio effective core potentials for molecular calculations. Potentials for the transition metal atoms Sc to Hg. *J. Chem. Phys.*, **1985**, 82 (1), 270–283. DOI: 10.1063/1.448799.

[42] Wadt, W. R.; Hay, P. Jeffrey. Ab initio effective core potentials for molecular calculations. Potentials for main group elements Na to Bi. *J. Chem. Phys.*, **1985**, 82 (1), 284–298. DOI: 10.1063/1.448800.

[43] Hay, P. Jeffrey; Wadt, W. R. Ab initio effective core potentials for molecular calculations. Potentials for K to Au including the outermost core orbitals. *J. Chem. Phys.*, **1985**, 82 (1), 299–310. DOI: 10.1063/1.448975.

[44] Ehlers, Andreas W.; Böhme, Mechthild; Dapprich, Stefan; Gobbi, Alberto; Höllwarth, Andreas; Jonas, Volker; Köhler, Karl Friedrich; Stegmann, Rainer; Veldkamp, Andre; Frenking, Gernot. A Set of f-Polarization Functions for Pseudo-Potential Basis Sets of the Transition Metals Sc–Cu, Y–Ag and La–Au. *Chem. Phys. Lett*, **1993**, 208 (1-2), 111–114. DOI: 10.1016/0009-2614(93)80086-5.

[45] Check, Catherine E.; Faust, Timothy O.; Bailey, John M.; Wright, Brian J.; Gilbert, Thomas M.; Sunderlin, Lee S. Addition of Polarization and Diffuse Functions to the LANL2DZ Basis Set for P-Block Elements. *J. Phys. Chem. A*, **2001**, 105 (34), 8111–8116. DOI: 10.1021/jp011945l.

[46] Roy, Lindsay E.; Hay, P. Jeffrey; Martin, Richard L. Revised Basis Sets for the LANL Effective Core Potentials. *J. Chem. Theory Comput.*, **2008**, 4 (7), 1029–1031. DOI: 10.1021/ct8000409.

[47] Papajak, Ewa; Truhlar, Donald G. Convergent Partially Augmented Basis Sets for Post-Hartree-Fock Calculations of Molecular Properties and Reaction Barrier Heights. *J. Chem. Theory Comput.*, **2011**, 7 (1), 10–18. DOI: 10.1021/ct1005533.

[48] Neese, F.; Valeev, E. F. Revisiting the Atomic Natural Orbital Approach for Basis Sets: Robust Systematic Basis Sets for Explicitly Correlated and Conventional Correlated ab initio Methods. *J. Chem. Theory Comput.*, **2011**, 7, 33–43.

[49] Müller, Marcel; Hansen, Andreas; Grimme, Stefan. ωB97X-3c: A composite range-separated hybrid DFT method with a molecule-optimized polarized valence double-ζ basis set. *J. Chem. Phys.*, **2023**, 158 (1), 014103. DOI: 10.1063/5.0133026.

[50] Pantazis, D. A.; Chen, X.-Y.; Landis, C. R.; Neese, F. *J. Chem. Theory Comput.*, **2008**, 4, 908–919.

[51] Bühl, M.; Reimann, C.; Pantazis, D. A.; Bredow, T.; Neese, F. Geometries of Third-Row Transition-Metal Complexes from Density-Functional Theory. *J. Chem. Theory Comput.*, **2008**, 4, 1449–1459. DOI: 10.1021/ct800172j.

[52] Pantazis, D. A.; Neese, F. *J. Chem. Theory Comput.*, **2009**, 5, 2229–2238.

[53] Pantazis, D. A.; Neese, F. *J. Chem. Theory Comput.*, **2011**, 7, 677–684.

[54] Pantazis, D. A.; Neese, F. *Theor. Chem. Acc.*, **2012**, 131, 1292.

[55] Rolfes, Julian D.; Neese, Frank; Pantazis, Dimitrios A. All-Electron Scalar Relativistic Basis Sets for the Elements Rb–Xe. *J. Comput. Chem.*, **2020**, 41, 1842–1849. DOI: 10.1002/jcc.26355.

[56] Aravena, D.; Neese, F.; Pantazis, Dimitrios A. Improved Segmented All-Electron Relativistically Contracted Basis Sets for the Lanthanides. *J. Chem. Theory Comput.*, **2016**, 12, 1148–1156. DOI: 10.1021/acs.jctc.5b01048.

[57] Pollak, Patrik; Weigend, Florian. Segmented Contracted Error-Consistent Basis Sets of Double- and Triple-ζ Valence Quality for One- and Two-Component Relativistic All-Electron Calculations. *J. Chem. Theory Comput.*, **2017**, 13 (8), 3696–3705. DOI: 10.1021/acs.jctc.7b00593.

[58] Franzke, Yannick J.; Treß, Robert; Pazdera, Tobias M.; Weigend, Florian. Error-Consistent Segmented Contracted All-Electron Relativistic Basis Sets of Double- and Triple-Zeta Quality for NMR Shielding Constants. *Phys. Chem. Chem. Phys.*, **2019**, 21 (30), 16658–16664. DOI: 10.1039/C9CP02382H.

[59] Stoychev, Georgi L.; Auer, Alexander A.; Neese, Frank. Automatic Generation of Auxiliary Basis Sets. *J. Chem. Theory Comput.*, **2017**, 13 (2), 554. DOI: 10.1021/acs.jctc.6b01041.

[60] Fuentealba, P.; Preuss, H.; Stoll, H.; von Szentpaly, L. *Chem. Phys. Lett.*, **1982**, 89, 418–422.

[61] von Szentpaly, L.; Fuentealba, P.; Preuss, H.; Stoll, H. *Chem. Phys. Lett.*, **1982**, 93, 555–559.

[62] Fuentealba, P.; Stoll, H.; von Szentpaly, L.; Schwerdtfeger, P.; Preuss, H. *J. Phys. B: At. Mol. Opt. Phys.*, **1983**, 16, L323.

[63] Stoll, H.; P. Fuentealba, P. Schwerdtfeger; Flad, J.; von Szentpaly, L.; Preuss, H. *J. Chem. Phys.*, **1984**, 81, 2732–2736.

[64] Fuentealba, P.; von Szentpaly, L.; Preuss, H.; Stoll, H. *J. Phys. B: At. Mol. Opt. Phys.*, **1985**, 18, 1287.

[65] Dolg, M.; Wedig, U.; Stoll, H.; Preuss, H. *J. Chem. Phys.*, **1987**, 86, 866–872.

[66] Igel-Mann, G.; Stoll, H.; Preuss, H. Pseudopotentials for Main Group Elements (IIIA through VIIA). *Mol. Phys.*, **1988**, 65 (6), 1321–1328. DOI: 10.1080/00268978800101811.

[67] Dolg, M.; Stoll, H.; Preuss, H. *J. Chem. Phys.*, **1989**, 90, 1730–1734.

[68] Schwerdtfeger, P.; Dolg, M.; Schwarz, W. H. E.; Bowmaker, G. A.; Boyd, P. D. W. *J. Chem. Phys.*, **1989**, 91, 1762–1774.

[69] Dolg, M.; Stoll, H.; Savin, A.; Preuss, H. *Theor. Chim. Acta*, **1989**, 75, 173–194.

[70] Andrae, D.; Häu\ssermann , U.; Dolg, M.; Stoll, H.; Preuss, H. Energy-Adjusted ab\textendashinitio Pseudopotentials for the Second and Third Row Transition Elements. *Theor. Chim. Acta*, **1990**, 77, 123–141. DOI: 10.1007/BF01114537.

[71] Kaupp, Martin; Schleyer, Paul v. R.; Stoll, Hermann; Preuss, Horst. Pseudopotential Approaches to Ca, Sr, and Ba Hydrides. Why Are Some Alkaline-Earth MX_2 Compounds Bent? *J. Chem. Phys.*, **1991**, 94 (2), 1360–1366. DOI: 10.1063/1.459993.

[72] Küchle, W.; Dolg, M.; Stoll, H.; Preuss, H. Energy-adjusted ab initio pseudopotentials for the rare earth elements. *Mol. Phys.*, **1991**, 74 (6), 1245–1263. DOI: 10.1063/1.456066.

[73] Dolg, M.; Fulde, P.; Küchle, W.; Neumann, C.-S.; Stoll, H. *J. Chem. Phys.*, **1991**, 94, 3011–3017.

[74] Dolg, M.; Stoll, H.; Flad, H.-J.; Preuss, H. *J. Chem. Phys.*, **1992**, 97, 1162–1173.

[75] Bergner, A.; Dolg, M.; Küchle, W.; Stoll, H.; Preuss, H. Ab initio energy-adjusted pseudopotentials for elements of groups 13–17. *Mol. Phys.*, **1993**, 80, 1431–1441. DOI: 10.1080/00268979300103121.

[76] Dolg, M.; Stoll, H.; Preuss, H.; Pitzer, R. M. *J. Phys. Chem.*, **1993**, 97, 5852–5859.

[77] Dolg, M.; Stoll, H.; Preuss, H. *Theor. Chim. Acta*, **1993**, 85, 441–450.

[78] Häuß ermann, U.; Dolg, M.; Stoll, H.; Preuss, H. *Mol. Phys.*, **1993**, 78, 1211–1224.

[79] Küchle, W.; Dolg, M.; Stoll, H.; Preuss, H. Energy-adjusted pseudopotentials for the actinides: Parameter sets and test calculations for thorium and thorium monoxide. *J. Chem. Phys.*, **1994**, 100 (10), 7535–7542. DOI: 10.1063/1.466847.

[80] Nicklass, A.; Dolg, M.; Stoll, H.; Preuss, H. *J. Chem. Phys.*, **1995**, 102, 8942–8952.

[81] Leininger, T.; Nicklass, A.; Stoll, H.; Dolg, M.; Schwerdtfeger, P. The accuracy of the pseudopotential approximation. II. A comparison of various core sizes for indium pseudopotentials in calculations for spectroscopic constants of InH, InF, and InCl. *J. Chem. Phys.*, **1996**, 105 (3), 1052–1059. DOI: 10.1063/1.471950.

[82] Leininger, T.; Nicklass, A.; Küchle, W.; Stoll, H.; Dolg, M.; Bergner, A. The accuracy of the pseudopotential approximation: Non-frozen-core effects for spectroscopic constants of alkali fluorides XF (X=K, Rb, Ca). *Chem. Phys. Lett.*, **1996**, 255 (4-6), 274–280. DOI: 10.1016/0009-2614(96)00382-X.

[83] Leininger, T.; Berning, A.; Nicklass, A.; Stoll, H.; Werner, H.-J.; Flad, H.-J. Spin-orbit interaction in heavy group 13 atoms and TlAr. *Chem. Phys.*, **1997**, 217 (1), 19–34. DOI: 10.1016/S0301-0104(97)00043-8.

[84] Schautz, F.; Flad, H.-J.; Dolg, M. *Theor. Chem. Acc.*, **1998**, 99, 231.

[85] Wang, Y.; Dolg, M. *Theor. Chem. Acc.*, **1998**, 100, 124.

[86] Metz, Bernhard; Stoll, Hermann; Dolg, Michael. Small-Core Multiconfiguration-Dirac–Hartree–Fock-Adjusted Pseudopotentials for Post-d Main Group Elements: Application to PbH and PbO. *J. Chem. Phys.*, **2000**, 113, 2563–2569. DOI: 10.1063/1.1305880.

[87] Metz, Bernhard; Schweizer, Markus; Stoll, Hermann; Dolg, Michael; Liu, Wenjian. Relativistic Energy-Consistent Pseudopotentials: Adjustments to Multi-Configuration Dirac–Hartree–Fock Data. *Theor. Chem. Acc.*, **2000**, 104, 22–28. DOI: 10.1007/s002140000187.

[88] Martin, Jan M. L.; Sundermann, Andreas. Correlation Consistent Valence Basis Sets for Use with the Stuttgart–Dresden–Bonn Relativistic Effective Core Potentials: The Atoms Ga–Kr and In–Xe. *J. Chem. Phys.*, **2001**, 114 (8), 3408–3420. DOI: 10.1063/1.1337864.

[89] Cao, X.; Dolg, M. Valence basis sets for relativistic energy-consistent small-core lanthanide pseudopotentials. *J. Chem. Phys.*, **2001**, 115, 7348. DOI: 10.1063/1.1406535.

[90] Stoll, H.; Metz, B.; Dolg, M. *J. Comput. Chem.*, **2002**, 23, 767.

[91] Cao, X.; Dolg, M. Segmented contraction scheme for small-core lanthanide pseudopotential basis sets. *J. Mol. Struct.: THEOCHEM*, **2002**, 581, 139. DOI: 10.1016/S0166-1280(01)00751-5.

[92] Peterson, K. A. *J. Chem. Phys.*, **2003**, 119, 11099.

[93] Peterson, K. A.; Figgen, D.; Goll, E.; Stoll, H.; Dolg, M. *J. Chem. Phys.*, **2003**, 119, 11113.

[94] Figgen, D.; Rauhut, G.; Dolg, M.; Stoll, H. *Chem. Phys.*, **2005**, 311, 227.

[95] Lim, I. S.; Schwerdtfeger, P.; Metz, B.; Stoll, H. All-electron and relativistic pseudopotential studies for the group 1 element polarizabilities from K to element 119. *J. Chem. Phys.*, **2005**, 122 (10), 104103. DOI: 10.1063/1.1856451.

[96] Peterson, K. A.; Puzzarini, C. *Theor. Chem. Acc.*, **2005**, 114, 283.

[97] Yang, J.; Dolg, M. *Theor. Chem. Acc.*, **2005**, 113, 212.

[98] Lim, I. S.; Stoll, H.; Schwerdtfeger, P. Relativistic small-core energy-consistent pseudopotentials for the alkaline-earth elements from Ca to Ra. *J. Chem. Phys.*, **2006**, 124 (3), 034107. DOI: 10.1063/1.2148945.

[99] Peterson, K. A.; Shepler, B. C.; Figgen, D.; Stoll, H. *J. Phys. Chem. A*, **2006**, 110, 13877.

[100] Peterson, K. A.; Figgen, D.; Dolg, M.; Stoll, H. *J. Chem. Phys.*, **2007**, 126, 124101.

[101] Moritz, Anna; Cao, Xiaoyan; Dolg, Michael. Quasirelativistic Energy-Consistent 5f-in-Core Pseudopotentials for Trivalent Actinide Elements. *Theor. Chem. Acc.*, **2007**, 117, 473–481. DOI: 10.1007/s00214-006-0180-7.

[102] Moritz, Anna; Dolg, Michael. Quasirelativistic Energy-Consistent 4f-in-Core Pseudopotentials for Trivalent Lanthanide Elements. *Theor. Chem. Acc.*, **2008**, 121, 297–304. DOI: 10.1007/s00214-008-0439-2.

[103] Hülsen, M.; Weigand, A.; Dolg, M. *Theor. Chem. Acc.*, **2009**, 122, 23.

[104] Figgen, D.; Peterson, K. A.; Dolg, M.; Stoll, H. *J. Chem. Phys.*, **2009**, 130, 164108.

[105] Weigand, A.; Cao, X.; Yang, J.; Dolg, M. *Theor. Chem. Acc.*, **2010**, 126, 117–127.

[106] Flores-Moreno, R.; Alvares-Mendez, R. J.; Vela, A.; Köster, A. M. *J. Comput. Chem.*, **2006**, 27, 1009.

[107] Giordano, L.; Pacchioni, G.; Bredow, T.; Sanz, J. F. *Surf. Sci.*, **2001**, 471, 21.

[108] Facility, Molecular Science Computing. Extensible Computational Chemistry Environment Basis Set Database. **2000**. URL: http://www.emsl.pnl.gov:2080/forms/basisform.html.

[109] Mitin, Alexander V.; Hirsch, Gernot; Buenker, Robert J. A Mean-Field Spin–Orbit Method Applicable to Correlated Wavefunctions. *Chem. Phys. Lett.*, **1996**, 259, 151–158. DOI: 10.1016/0009-2614(96)00119-4.

[110] Mitin, Alexander V.; Hirsch, Gernot; Buenker, Robert J. A Mean-Field Spin–Orbit Method Applicable to Correlated Wavefunctions. *J. Comput. Chem.*, **1997**, 18, 1200–1210. DOI: 10.1002/(SICI)1096-987X(19970715)18:9<1200::AID-JCC8>3.0.CO;2-2.

[111] Whitten, J. L. *J. Chem. Phys.*, **1973**, 58, 4496. DOI: 10.1063/1.1679012.

[112] Baerends, E. J.; Ellis, D. E.; Ros, P. Self-consistent molecular Hartree—Fock—Slater calculations I. The computational procedure. *Chem. Phys.*, **1973**, 2, 41. DOI: 10.1016/0301-0104(73)80059-X.

[113] Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *J. Chem. Phys.*, **1979**, 71, 3396.

[114] Van Alsenoy, C. *J. Comput. Chem.*, **1988**, 9, 620.

[115] Kendall, Rick A.; Früchtl, Herbert A. The Impact of the Resolution of the Identity Approximate Integral Method on Modern Ab Initio Algorithm Development. *Theor. Chem. Acc.*, **1997**, 97, 158–163. DOI: 10.1007/s002140050249.

[116] Eichkorn, K.; Treutler, O.; Öhm, H.; Häser, M.; Ahlrichs, R. *Chem. Phys. Lett.*, **1995**, 240, 283.

[117] Eichkorn, K.; Weigend, F.; Treutler, O.; Ahlrichs, R. *Theor. Chem. Acc.*, **1997**, 97, 119.

[118] Vahtras, O.; Almlöf, J.; Feyereisen, M. W. *Chem. Phys. Lett.*, **1993**, 213, 514.

[119] Neese, F.; Wennmohs, F.; Hansen, A.; Becker, U. *Chem. Phys.*, **2009**, 356, 98–109.

[120] Helmich-Paris, Benjamin; de Souza, Bernardo; Neese, Frank; Izsák, Róbert. An improved chain of spheres for exchange algorithm. *J. Chem. Phys.*, **2021**, 155 (10), 104109. DOI: 10.1063/5.0058766.

[121] Helmich-Paris, Benjamin. A trust-region augmented Hessian implementation for state-specific and state-averaged CASSCF wave functions. *J. Chem. Phys.*, **2022**, 156 (20), 204104. arXiv:10.1063/5.0090447, DOI: 10.1063/5.0090447.

[122] Kossmann, Simone; Neese, Frank. Comparison of two efficient approximate Hartee–Fock approaches. *Chem. Phys. Lett.*, **2009**, 481 (4-6), 240–243. DOI: 10.1016/j.cplett.2009.10.007.

[123] Treutler, O.; Ahlrichs, R. J. *J. Chem. Phys.*, **1994**, 102, 346.

[124] Kruse, Holger; Grimme, Stefan. A geometrical correction for the inter- and intra-molecular basis set superposition error in Hartree-Fock and density functional theory calculations for large systems. *J. Chem. Phys.*, **2012**, 136 (15), 154101. DOI: 10.1063/1.3700154.

[125] Kruse, Holger; Goerigk, Lars; Grimme, Stefan. Why the Standard B3LYP/6-31G* Model Chemistry Should Not Be Used in DFT Calculations of Molecular Thermochemistry: Understanding and Correcting the Problem. *J. Org. Chem.*, **2012**, 77 (23), 10824–10834. DOI: 10.1021/jo301927e.

[126] van Lenthe, E.; Baerends, E. J.; Snijders, J. G. *J. Chem. Phys.*, **1994**, 101, 9783–9792.

[127] van Wüllen, C. *J. Chem. Phys.*, **1998**, 109, 392–399.

[128] Sandhoefer, B.; Neese, F. One-electron contributions to the g-tensor for second-order Douglas–Kroll–Hess theory. *J. Chem. Phys.*, **2012**, 137, 094102.

[129] Peng, Daoling; Middendorf, Nils; Weigend, Florian; Reiher, Markus. An efficient implementation of two-component relativistic exact-decoupling methods for large molecules. *J. Chem. Phys.*, **2013**, 138 (18), 184105. DOI: 10.1063/1.4803693.

[130] Franzke, Yannick J.; Middendorf, Nils; Weigend, Florian. Efficient implementation of one- and two-component analytical energy gradients in exact two-component theory. *J. Chem. Phys.*, **2018**, 148 (10), 104110. DOI: 10.1063/1.5022153.

[131] Franzke, Yannick J.; Yu, Jason M. Hyperfine Coupling Constants in Local Exact Two-Component Theory. *J. Chem. Theory Comput.*, **2022**, 18 (1), 323–343. DOI: 10.1021/acs.jctc.1c01027.

[132] Franzke, Yannick J.; Mack, Fabian; Weigend, Florian. NMR Indirect Spin–Spin Coupling Constants in a Modern Quasi-Relativistic Density Functional Framework. *J. Chem. Theory Comput.*, **2021**, 17 (7), 3974–3994. DOI: 10.1021/acs.jctc.1c00167.

[133] Franzke, Yannick J; Weigend, Florian. NMR Shielding Tensors and Chemical Shifts in Scalar-Relativistic Local Exact Two-Component Theory. *J. Chem. Theory Comput.*, **2019**, 15 (2), 1028–1043. DOI: 10.1021/acs.jctc.8b01084.

[134] Cheng, Lan; Gauss, Jürgen. Analytic energy gradients for the spin-free exact two-component theory using an exact block diagonalization for the one-electron Dirac Hamiltonian. *J. Chem. Phys.*, **2011**, 135 (8), 084114. DOI: 10.1063/1.3624397.

[135] Cheng, Lan; Gauss, Jürgen. Analytic second derivatives for the spin-free exact two-component theory. *J. Chem. Phys.*, **2011**, 135 (24), 244104. DOI: 10.1063/1.3667202.

[136] Cheng, Lan; Gauss, Jürgen; Stanton, John F. Treatment of scalar-relativistic effects on nuclear magnetic shieldings using a spin-free exact-two-component approach. *J. Chem. Phys.*, **2013**, 139 (5), 054105. DOI: 10.1063/1.4816130.

[137] Peng, Daoling; Reiher, Markus. Local relativistic exact decoupling. *J. Chem. Phys.*, **2012**, 136 (24), 244108. DOI: 10.1063/1.4729788.

[138] Visscher, L.; Dyall, K. G. *Atom. Data Nucl. Data Tabl.*, **1997**, 67, 207.

[139] Barone, V.; Cossi, M. Quantum Calculation of Molecular Energies and Energy Gradients in Solution by a Conductor Solvent Model. *J. Phys. Chem. A*, **1998**, 102, 1995–2001. DOI: 10.1021/jp9716997.

[140] Garcia-Ratés, M.; Neese, F. Effect of the Solute Cavity on the Solvation Energy and its Derivatives within the Framework of the Gaussian Charge Scheme. *J. Comput. Chem.*, **2020**, 41 (9), 922–939. DOI: 10.1002/jcc.26139.

[141] Marenich, Aleksandr V.; Cramer, Christopher J.; Truhlar, Donald G. Universal Solvation Model Based on Solute Electron Density and on a Continuum Model of the Solvent Defined by the Bulk Dielectric Constant and Atomic Surface Tensions. *J. Phys. Chem. B*, **2009**, 113 (18), 6378–6396. DOI: 10.1021/jp810292n.

[142] Gerlach, Thomas; Müller, Simon; González de Castilla, Andrés; Smirnova, Irina. An Open Source COSMO-RS Implementation and Parameterization Supporting the Efficient Implementation of Multiple Segment Descriptors. *Fluid Phase Equil.*, **2022**, 560, 113472.

[143] York, D. M.; Karplus, M. *J. Phys. Chem. A*, **1999**, 103, 11060–11079.

[144] Pascual-Ahuir, J. L.; Silla, E. *J. Comput. Chem.*, **1990**, 11, 1047–1060.

[145] Pascual-Ahuir, J. L.; Silla, E.; Tunon, I. *J. Comput. Chem.*, **1991**, 12, 1077–1088.

[146] Pascual-Ahuir, J. L.; Silla, E.; Tunon, I. *J. Comput. Chem.*, **1994**, 15, 1127–1138.

[147] Lange, Adrian W.; Herbert, John M. A smooth, nonsingular, and faithful discretization scheme for polarizable continuum models: The switching/Gaussian approach. *J. Chem. Phys.*, **2010**, 133 (24), 244111. DOI: 10.1063/1.3511297.

[148] Haynes, W. M.; Lide, D. R.; Bruno, T. J. *Handbook of Chemistry and Physics*. CRC Press, 95th edition, **2014**. ISBN 978-1-4822-0868-9.

[149] Stahn, M.; Ehlert, S.; Grimme, S. Extended Conductor-like Polarizable Continuum Solvation Model (CPCM-X) for Semiempirical Methods. *J. Phys. Chem. A*, **2023**, 127, 7036–7043. DOI: 10.1021/acs.jpca.3c04382.

[150] Bondi, A. *J. Phys. Chem.*, **1964**, 68, 441–451.

[151] Mantina, M.; Chamberlin, A. C.; Valero, R.; Cramer, C. J.; Truhlar, D. G. *J. Phys. Chem. A*, **2009**, 113, 5806–5812.

[152] Klamt, A.; Eckert, F. COSMO-RS: a novel and efficient method for the a priori prediction of thermophysical data of liquids. *Fluid Phase Equil.*, **2000**, 172, 43–72. DOI: 10.1016/S0378-3812(00)00357-5.

[153] Truong, T. N.; Stefanovich, E. V. *Chem. Phys. Lett.*, **1995**, 240, 253–260.

[154] Pierotti, R. A. *Chem. Rev.*, **1976**, 76, 717.

[155] Claverie, P.; Daudey, J. P.; Langlet, J.; Pullman, B.; Plazzola, D.; Huron, M. J. Studies of solvent effects. 1. Discrete, continuum, and discrete –continuum models and their comparison for some simple cases: ammonium(1+) ion, methanol, and substituted ammonium(1+) ion. *J. Phys. Chem.*, **1978**, 82, 405–418. DOI: 10.1021/j100493a008.

[156] Pye, C. C.; Ziegler, T. *Theor. Chem. Acc.*, **1999**, 101, 396.

[157] Engelage, Elric.; Schulz, Nils; Flemming, Heinen; Huber, Stefan M.; Truhlar, Donald G.; Cramer, Christopher J. Refined SMD Parameters for Bromine and Iodine Accurately Model Halogen-Bonding Interactions in Solution. *Chem. Eur. J.*, **2018**, 24, 15983.

[158] Plett, Christoph; Stahn, Marcel; Bursch, Markus; Mewes, Jan-Michael; Grimme, Stefan. Improving Quantum Chemical Solvation Models by Dynamic Radii Adjustment for Continuum Solvation Method (DRACO). *J. Phys. Chem. Lett.*, **2024**, 15, 2462.

[159] Caldeweyher, Eike; Ehlert, Sebastian; Hansen, Andreas; Neugebauer, Hagen; Spicher, Sebastian; Bannwarth, Christoph; Grimme, Stefan. A Generally Applicable Atomic-Charge Dependent London Dispersion Correction. *J. Chem. Phys.*, **2019**, 150 (15), 154122. arXiv:10.1063/1.5090222, DOI: 10.1063/1.5090222.

[160] Müller, Marcel; Hansen, Andreas; Grimme, Stefan. An atom-in-molecule adaptive polarized valence single-ζ atomic orbital basis for electronic structure calculations. *J. Chem. Phys.*, **2023**, 159, 164108.

[161] Bannwarth, Christoph; Ehlert, Sebastian; Grimme, Stefan. GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1652–1671. DOI: 10.1021/acs.jctc.8b01176.

[162] Wittmann, Lukas; Garcia-Ratés, Miquel; Riplinger, Christoph. Analytical First Derivatives of the SCF Energy for the Conductor-like Polarizable Continuum Model with Non-Static Radii. J. Comput. Chem., **2025**. DOI: 10.1002/jcc.70099.

[163] Müller, Simon; Nevolianis, Thomas; Garcia-Ratés, Miquel; Riplinger, Christoph; Leonhard, Kai; Smirnova, Irina. Predicting solvation free energies for neutral molecules in any solvent with openCOSMO-RS. *Fluid Phase Equilibria*, **2025**, 589, 114250. DOI: 10.1016/j.fluid.2024.114250.

[164] Klamt, Andreas. Conductor-like Screening Model for Real Solvents: A New Approach to the Quantitative Calculation of Solvation Phenomena. *J. Phys. Chem.*, **1995**, 99, 2224–2235.

[165] Klamt, Andreas; Volker, Jonas; Bürger, Thorsten; Lohrenz, John C. W. Refinement and Parametrization of COSMO-RS. *J. Phys. Chem. A*, **1998**, 102, 5074–5085.

[166] Cammi, R. Quantum cluster theory for the polarizable continuum model. I. The CCSD level with analytical first and second derivatives. *J. Chem. Phys*, **2009**, 131, 164104. DOI: 10.1063/1.3245400.

[167] Caricato, M. CCSD-PCM: Improving upon the reference reaction field approximation at no cost. *J. Chem. Phys*, **2011**, 135, 074113. DOI: 10.1063/1.3624373.

[168] Garcia-Ratés, M.; Becker, U.; Neese, F. Implicit Solvation in Domain Based Pair Natural Orbital Coupled Cluster (DLPNO-CCSD) Theory. *J. Comput. Chem.*, **2021**, 42 (27), 1959–1973. DOI: 10.1002/jcc.26726.

[169] Almlöf, J. Faegri, K.; Korsell, K. Principles for a direct SCF approach to LICAO-MO ab-initio calculations. *J. Comput. Chem.*, **1982**, 3, 385. DOI: 10.1002/jcc.540030314.

[170] Almlöf, J.; Taylor, P. R. Computational Aspects of Direct SCF and MCSCF Methods. In Dykstra, C. E., editor, *Advanced Theories and Computational Approaches to the Electronic Structure of Molecules*, pages 107. Springer, **1984**. DOI: 10.1007/97S-94-009-6451-S.

[171] Almlöf, J. Direct Methods in Electronic Structure Theory. In Yarkony, D. R., editor, *Modern Electronic Structure Theory*, pages 110. World Scientific, **1995**. DOI: 10.1142/1957.

[172] Häser, M.; Ahlrichs, R. *J. Comput. Chem.*, **1989**, 10, 104.

[173] Seeger, R.; Pople, J. A. *J. Chem. Phys.*, **1977**, 66, 3045.

[174] Bauernschmitt, R.; Ahlrichs, R. Stability analysis for solutions of the closed shell Kohn\textendashSham equation. *J. Chem. Phys.*, **1996**, 104, 9047. DOI: 10.1063/1.471637.

[175] Shaik, Sason; Ramanan, Rajeev; Danovich, David; Mandal, Debasish. Structure and reactivity/selectivity control by oriented-external electric fields. *Chem. Soc. Rev.*, **2018**, 47, 5125–5145. DOI: 10.1039/C8CS00354H.

[176] Mulliken, R. S. Report on Notation for the Spectra of Polyatomic Molecules. *J. Chem. Phys.*, **1955**, 23 (11), 1997–2011. DOI: 10.1063/1.1740655.

[177] Schutte, C. J. H.; Bertie, J. E.; Bunker, P. R.; Hougen, J. T.; Mills, I. M.; Watson, J. K. G.; Winnewisser, B. P. Notations and conventions in molecular spectroscopy: Part 2. Symmetry notation (IUPAC Recommendations 1997). *Pure & Appl. Chem.*, **1997**, 69 (8), 1641–1649. DOI: 10.1351/pac199769081641.

[178] Amos, A. T.; Hall, G. G. Single determinant wave functions. *Proc. R. Soc. Ser. A.*, **1961**, 263, 483. DOI: 10.1098/rspa.1961.0175.

[179] King, Harry F.; Stanton, Richard E.; Kim, Hojing; Wyatt, Robert E.; Parr, Robert G. Corresponding Orbitals and the Nonorthogonality Problem in Molecular Quantum Mechanics. *J. Chem. Phys.*, **1967**, 47, 1936–1941. DOI: 10.1063/1.1712221.

[180] Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover Publications, **1989**. ISBN 978-0-486-69186-2.

[181] Dewar, M. J. S.; Hashmall, J. A.; Venier, C. G. *J. Am. Chem. Soc.*, **1968**, 90, 1953.

[182] McWeeny, R. SCF Theory for Excited States. I. *Mol. Phys.*, **1974**, 28 (5), 1273–1282. DOI: 10.1080/00268977400102581.

[183] Brobowicz, F. W.; Goddard, W. A. In III, H. F. Schaefer, editor, *Methods of Electronic Structure Theory*, pages 79. Plenum Press, **1977**.

[184] Carbo, R.; Riera, J. M. *A General SCF Theory. Lecture Notes in Chemistry*. Springer Verlag, **1978**.

[185] Binkley, J. S.; Pople, J. A.; Dobosh, P. A. The calculation of spin-restricted single-determinant wavefunctions. *Mol. Phys.*, **1974**, 28, 1423. DOI: 10.1080/00268977400102701.

[186] Edwards, W. D.; Zerner, M. C. *Theor. Chim. Acta*, **1987**, 72, 347.

[187] Muller, R. P.; Langlois, J. M.; Ringnalda, M. N.; Friesner, R. A.; Goddard, W. A. A Generalized Direct Inversion in the Iterative Subspace Approach for Generalized Valence Bond Wavefunctions. *J. Chem. Phys.*, **1994**, 100, 1226–1239. DOI: 10.1063/1.466639.

[188] Bofill, J. M.; Bono, H.; Rubio, J. Analysis of the convergence of the general coupling operator method for one-configuration-type wave functions. *J. Comput. Chem.*, **1998**, 19, 368. DOI: 10.1002/(SICI)1096-987X(199802)19:3<368::AID-JCC10>3.0.CO;2-E.

[189] Stavrev, K. K.; Zerner, M. C. Spin-averaged Hartree–Fock procedure for spectroscopic calculations: The absorption spectrum of Mn2+ in ZnS crystals. *Int. J. Quant. Chem.*, **1997**, 65 (5), 877–884. DOI: 10.1002/(SICI)1097-461X(1997)65:5<877::AID-QUA51>3.0.CO;2-T.

[190] Zerner, M. C. *Int. J. Quant. Chem.*, **1989**, 35, 567.

[191] Leyser da Costa Gouveia, Tiago; Maganas, Dimitrios; Neese, Frank. Restricted Open-Shell Hartree–Fock Method for a General Configuration State Function Featuring Arbitrarily Complex Spin-Couplings. *J. Phys. Chem. A*, **2024**, 128 (25), 5041–5053. PMID: 38886177. DOI: 10.1021/acs.jpca.4c00688.

[192] Hartree, D. R. The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part I. Theory and Methods. *Proc. Cambridge Phil. Soc.*, **1928**, 24, 89–110. DOI: 10.1017/S0305004100011919.

[193] Slater, J. C. The Theory of Complex Spectra. *Phys. Rev.*, **1929**, 34, 1293–1322. DOI: 10.1103/PhysRev.34.1293.

[194] Fock, V. Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems. *Z. Phys.*, **1930**, 61, 126–148. DOI: 10.1007/BF01340294.

[195] Perdew, John P.; Schmidt, Karla. Jacob's ladder of density functional approximations for the exchange-correlation energy. *AIP Conf. Proc.*, **2001**, 577 (1), 1–20. DOI: 10.1063/1.1390175.

[196] Vosko, S. H.; Wilk, L.; Nusair, M. *Can. J. Phys.*, **1980**, 58, 1200.

[197] Perdew, J. P.; Wang, Y. *Phys. Rev. B*, **1992**, 45, 13244.

[198] Becke, A. D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, **1988**, 38, 3098. DOI: 10.1103/PhysRevA.38.3098.

[199] Perdew, J. P. *Phys. Rev. B*, **1986**, 33, 8822.

[200] Miehlich, Burkhard; Savin, Andreas; Stoll, Hermann; Preuss, Heinzwerner. Results obtained with the correlation energy density functionals of becke and Lee, Yang and Parr. *Chem. Phys. Lett*, **1989**, 157 (3), 200–206. DOI: 10.1016/0009-2614(89)87234-3.

[201] Handy, Nicholas C.; Cohen, Aron J. Left-right correlation energy. *Mol. Phys.*, **2001**, 99 (5), 403–412. DOI: 10.1080/00268970010018431.

[202] Gill, P. M. W. *Mol. Phys.*, **1996**, 89, 433. DOI: 10.1080/002689796173813.

[203] Xu, Xin; Goddard, III, William A. *Proc. Nat. Acad. Sci.*, **2004**, 101, 2673.

[204] Germany), W. E. Heraeus Seminar (75th 1991 Gaussig. *Electronic structure of solids '91: proceedings of the 75. WE-Heraeus-Seminar and 21st Annual International Symposium on Electronic Structure of Solids held in Gaussig (Germany), March 11-15, 1991*. Akademie Verlag, Berlin, 1st ed. edition, **1991**. ISBN 978-3-05-501504-5. Open Library ID: OL1778449M.

[205] Adamo, C.; Barone, V. Exchange functionals with improved long-range behavior and adiabatic connection methods without adjustable parameters: The mPW and mPW1PW models. *J. Chem. Phys.*, **1998**, 108, 664. DOI: 10.1063/1.475428.

[206] Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.*, **1996**, 77, 3865.

[207] Hammer, B.; Hansen, L. B.; Nørskov, J. K. *Phys. Rev. B*, **1999**, 59, 7413.

[208] Zhang, Y.; Yang, W. *Phys. Rev. Lett.*, **1998**, 80, 890.

[209] Murray, Éamonn D.; Lee, Kyuho; Langreth, David C. Investigation of Exchange Energy Density Functional Accuracy for Interacting Molecules. *J. Chem. Theory Comput.*, **2009**, 5 (10), 2754–2762. DOI: 10.1021/ct900365q.

[210] Brandenburg, Jan Gerit; Bannwarth, Christoph; Hansen, Andreas; Grimme, Stefan. B97-3c: A Revised Low-Cost Variant of the B97-D Density Functional Method. *J. Chem. Phys.*, **2018**, 148 (6), 064104. DOI: 10.1063/1.5012601.

[211] Mardirossian, Narbe; Head-Gordon, Martin. Mapping the genome of meta-generalized gradient approximation density functionals: The search for B97M-V. *J. Chem. Phys.*, **2015**, 142, 074111. DOI: 10.1063/1.4907719.

[212] Najibi, A.; Goerigk, L. *J. Chem. Theory Comput.*, **2018**, 14, 5725.

[213] Najibi, A.; Goerigk, L. DFT-D4 Counterparts of Leading Meta-Generalized-Gradient Approximation and Hybrid Density Functionals for Energetics and Geometries. *J. Comput. Chem.*, **2020**, 41, 2562–2572. DOI: 10.1002/jcc.26411.

[214] Sun, Jianwei; Ruzsinszky, Adrienn; Perdew, John P. Strongly Constrained and Appropriately Normed Semilocal Density Functional. *Phys. Rev. Lett.*, **2015**, 115 (3), 036402. DOI: 10.1103/PhysRevLett.115.036402.

[215] Bartók, Albert P; Yates, Jonathan R. Regularized SCAN Functional. *J. Chem. Phys.*, **2019**, 150 (16), 161101. DOI: 10.1063/1.5094646.

[216] Furness, James W; Kaplan, Aaron D; Ning, Jinliang; Perdew, John P; Sun, Jianwei. Accurate and Numerically Efficient r^2SCAN Meta-Generalized Gradient Approximation. *J. Phys. Chem. Lett.*, **2020**, 11 (19), 8208–8215. DOI: 10.1021/acs.jpclett.0c02405.

[217] Zhao, Yan; Truhlar, Donald G. A new local density functional for main-group thermochemistry, transition metal bonding, thermochemical kinetics, and noncovalent interactions. *J. Chem. Phys.*, **2006**, 125 (19), 194101. DOI: 10.1063/1.2370993.

[218] Staroverov, V. N.; Scuseria, G. E.; Tao, J.; Perdew, J. P. *J. Chem. Phys.*, **2003**, 119, 12129.

[219] Perdew, John P.; Ruzsinszky, Adrienn; Csonka, Gábor I.; Constantin, Lucian A.; Sun, Jianwei. Workhorse Semilocal Density Functional for Condensed Matter Physics and Quantum Chemistry. *Phys. Rev. Lett.*, **2009**, 103, 026403. DOI: 10.1103/PhysRevLett.103.026403.

[220] Perdew, John P.; Ruzsinszky, Adrienn; Csonka, Gábor I.; Constantin, Lucian A.; Sun, Jianwei. Erratum: Workhorse Semilocal Density Functional for Condensed Matter Physics and Quantum Chemistry [Phys. Rev. Lett. 103, 026403 (2009)]. *Phys. Rev. Lett.*, **2011**, 106, 179902. DOI: 10.1103/PhysRevLett.106.179902.

[221] Grimme, Stefan; Hansen, Andreas; Ehlert, Sebastian; Mewes, Jan-Michael. r2SCAN-3c: A "Swiss Army Knife" Composite Electronic-Structure Method. *J. Chem. Phys.*, **2021**, 154 (6), 064103. arXiv:10.1063/5.0040021, DOI: 10.1063/5.0040021.

[222] Ahlrichs, R.; Bär, M.; Baron, H. P.; Bauernschmitt, R.; Böcker, S.; Ehrig, M.; Eichkorn, K.; Elliott, S.; Furche, F.; Haase, F.; Häser, M.; Horn, H.; Huber, C.; Huniar, U.; Kattanek, M.; Kölmel, C.; Kollwitz, M.; May, K.; Ochsenfeld, C.; Öhm, H.; Schäfer, A.; Schneider, U.; Treutler, O.; von Arnim, M.; Weigend, F.; Weis, P.; Weiss, H. *TurboMole - Program System for Ab Initio Electronic Structure Calculations, Version 5.2*. Universität Karlsruhe, Karlsruhe, Germany, **2000**.

[223] Ahlrichs, R. In Schleyer, P. v. R., editor, *Encyclopedia of Computational Chemistry*, pages 3123. John Wiley and Sons, **1998**. DOI: 10.1002/0470845015.

[224] Ahlrichs, R.; Bär, M.; Häser, M.; Horn, H.; Kölmel, C. Electronic structure calculations on workstation computers: The program system turbomole. *Chem. Phys. Lett.*, **1989**, 162, 165. DOI: 10.1016/0009-2614(89)85118-8.

[225] Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Zakrzewski, V. G.; Montgomery, Jr., J. A.; Stratmann, R. E.; Burant, J. C.; Dapprich, S.; Millam, J. M.; Daniels, A. D.; Kudin, K. N.; Strain, M. C.; Farkas, O.; Tomasi, J.; Barone, V.; Cossi, M.; Cammi, R.; Mennucci, B.; Pomelli, C.; Adamo, C.; Clifford, S.; Ochterski, J.; Petersson, G. A.; Ayala, P. Y.; Cui, Q.; Morokuma, K.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Cioslowski, J.; Ortiz, J. V.; Baboul, A. G.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Gomperts, R.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Challacombe, M.; Gill, P. M. W.; Johnson, B.; Chen, W.; Wong, M. W.; Andres, J. L.; Gonzalez, C.; Head-Gordon, M.; Replogle, E. S.; Pople, J. A. *Gaussian 98, Revision A.8*. Gaussian, Inc., Pittsburgh PA, **1998**.

[226] Lee, C.; Yang, W.; Parr, R. G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, **1988**, 37 (2), 785–789. DOI: 10.1103/PhysRevB.37.785.

[227] Becke, A. D. A new mixing of Hartree-Fock and local density-functional theories. *J. Chem. Phys.*, **1993**, 98, 1372. DOI: 10.1063/1.464304.

[228] Becke, A. D. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.*, **1993**, 98, 5648. DOI: 10.1063/1.464913.

[229] Adamo, C.; Barone, V. Toward Reliable Density Functional Methods without Adjustable Parameters: The PBE0 Model. *J. Chem. Phys.*, **1999**, 110, 6158. DOI: 10.1063/1.478522.

[230] Stephens, P. J.; Devlin, F. J.; Chabalowski, C. F.; Frisch, M. J. Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields. *J. Phys. Chem.*, **1994**, 98, 11623. DOI: 10.1021/j100096a001.

[231] Ernzerhof, M. In Joubert, D. P., editor, *Density Functionals: Theory and Applications*. Springer Verlag, **1998**.

[232] Adamo, C.; Barone, V. Toward reliable adiabatic connection models free from adjustable parameters. *Chem. Phys. Lett*, **1997**, 274 (1), 242–250. DOI: 10.1016/S0009-2614(97)00651-9.

[233] aron J. Cohen; Handy, Nicholas C. Dynamic correlation. *Mol. Phys.*, **2001**, 99 (7), 607–615. DOI: 10.1080/00268970010023435.

[234] Goerigk, Lars; Grimme, Stefan. A thorough benchmark of density functional methods for general main group thermochemistry, kinetics, and noncovalent interactions. *Phys. Chem. Chem. Phys.*, **2011**, 13, 6670–6688. DOI: 10.1039/C0CP02984J.

[235] Zhao, Yan; Truhlar, Donald G. The M06 suite of density functionals for main group thermochemistry, thermochemical kinetics, noncovalent interactions, excited states, and transition elements: two new functionals and systematic testing of four M06-class functionals and 12 other functionals. *Theor. Chem. Acc.*, **2008**, 120 (1-3), 215–241. DOI: 10.1007/s00214-007-0310-x.

[236] Zhao, Y.; Truhlar, D. G. *J. Phys. Chem. A*, **2005**, 109, 5656–5667.

[237] Grimme, Stefan. Accurate Calculation of the Heats of Formation for Large Main Group Compounds with Spin-Component Scaled MP2 Methods. *J. Phys. Chem. A*, **2005**, 109 (13), 3067–3077. DOI: 10.1021/jp050036j.

[238] Bursch, Markus; Neugebauer, Hagen; Ehlert, Sebastian; Grimme, Stefan. Dispersion corrected r^2SCAN based global hybrid functionals: r^2SCANh, r^2SCAN0, and r^2SCAN50. *J. Chem. Phys.*, **2022**, 156 (13), 134105. DOI: 10.1063/5.0086040.

[239] Grimme, S.; Brandenburg, J. G.; Bannwarth, C.; Hansen, A. Consistent structures and interactions by density functional theory with small atomic orbital basis sets. *J. Chem. Phys.*, **2015**, 143, 054107. DOI: 10.1063/1.4927476.

[240] Pracht, Philipp; Grant, David F.; Grimme, Stefan. Comprehensive Assessment of GFN Tight-Binding and Composite Density Functional Theory Methods for Calculating Gas-Phase Infrared Spectra. *J. Chem. Theory Comput.*, **2020**, 16 (11), 7044–7060. DOI: 10.1021/acs.jctc.0c00877.

[241] Iikura, H.; Tsuneda, T.; Yanai, T.; Hirao, K. A long-range correction scheme for generalized-gradient-approximation exchange functionals. *J. Chem. Phys.*, **2001**, 115, 3540–3544. DOI: 10.1063/1.1383587.

[242] Yanai, T.; Tew, D. P.; Handy, N. C. *Chem. Phys. Lett.*, **2004**, 393, 51–57.

[243] Chai, J.-D.; Head-Gordon, M. *J. Chem. Phys.*, **2008**, 128, 084106.

[244] Mardirossian, Narbe; Head-Gordon, Martin. ωB97X-V: A 10-parameter, range-separated hybrid, generalized gradient approximation density functional with nonlocal correlation, designed by a survival-of-the-fittest strategy. *Phys. Chem. Chem. Phys.*, **2014**, 16 (21), 9904–9924. DOI: 10.1039/C3CP54374A.

[245] Lin, Y.-S.; Li, G.-D.; Mao, S.-P.; Chai, J.-D. Long-Range Corrected Hybrid Density Functionals with Improved Dispersion Corrections. *J. Chem. Theory Comput.*, **2013**, 9, 263–272. DOI: 10.1021/ct300715s.

[246] Tawada, Y.; Tsuneda, T.; Yanagisawa, S.; Yanai, T.; Hirao, K. *J. Chem. Phys.*, **2004**, 120, 8425–8433.

[247] Mardirossian, Narbe; Head-Gordon, Martin. ωB97M-V: A combinatorially optimized, range-separated hybrid, meta-GGA density functional with VV10 nonlocal correlation. *J. Chem. Phys.*, **2016**, 144, 214110. DOI: 10.1063/1.4952647.

[248] Wittmann, Lukas; Neugebauer, Hagen; Grimme, Stefan; Bursch, Markus. Dispersion-corrected r^2SCAN based double-hybrid functionals. *J. Chem. Phys.*, **2023**, 159 (22), 224103. DOI: 10.1063/5.0174988.

[249] Grimme, S. *J. Chem. Phys.*, **2006**, 124, 034108.

[250] Goerigk, L.; Grimme, S. *J. Chem. Theory Comput.*, **2011**, 7, 291–309.

[251] Kozuch, Sebastian; Gruzman, David; Martin, Jan M. L. DSD-BLYP: A General Purpose Double Hybrid Density Functional Including Spin Component Scaling and Dispersion Correction. *J. Phys. Chem. C*, **2010**, 114 (48), 20801–20808. DOI: 10.1021/jp1070852.

[252] Kozuch, Sebastian; Martin, Jan M. L. DSD-PBEP86: In Search of the Best Double-Hybrid DFT with Spin-Component Scaled MP2 and Dispersion Corrections. *Phys. Chem. Chem. Phys.*, **2011**, 13 (45), 20104–20107. DOI: 10.1039/C1CP22592H.

[253] Kozuch, Sebastian; Martin, Jan M. L. Spin-component-scaled Double Hybrids: An Extensive Search for the Best Fifth-rung Functionals Blending DFT and Perturbation Theory. *J. Comput. Chem.*, **2013**, 34 (27), 2327–2344. DOI: 10.1002/jcc.23391.

[254] Santra, Golokesh; Cho, Minsik; Martin, Jan M. L. Exploring Avenues beyond Revised DSD Functionals: I. Range Separation, with xDSD as a Special Case. *J. Phys. Chem. A*, **2021**, 125 (21), 4614–4627. DOI: 10.1021/acs.jpca.1c01294.

[255] Zhang, Ying; Xu, Xin; Goddard, William A. Doubly hybrid density functional for accurate descriptions of nonbond interactions, thermochemistry, and thermochemical kinetics. *Proc. Natl. Acad. Sci. USA*, **2009**, 106 (13), 4963–4968. DOI: 10.1073/pnas.0901093106.

[256] Mardirossian, Narbe; Head-Gordon, Martin. Survival of the most transferable at the top of Jacob's ladder: Defining and testing the ωB97M(2) double hybrid density functional. *J. Chem. Phys.*, **2018**, 148 (24), 241736. DOI: 10.1063/1.5025226.

[257] Neugebauer, Hagen; Pinski, Peter; Grimme, Stefan; Neese, Frank; Bursch, Markus. Assessment of DLPNO-MP2 Approximations in Double-Hybrid DFT. *J. Chem. Theory Comput.*, **2023**, 19 (21), 7695–7703. DOI: 10.1021/acs.jctc.3c00896.

[258] Neese, F.; Schwabe, T.; Grimme, S. *J. Chem. Phys.*, **2007**, 126, 124115.

[259] Najibi, A.; Casanova-Páez, M.; Goerigk, L. Analysis of Recent BLYP- and PBE-Based Range-Separated Double-Hybrid Density Functional Approximations for Main-Group Thermochemistry, Kinetics, and Noncovalent Interactions. *J. Phys. Chem. A*, **2021**, 125, 4026–4035.

[260] Schwabe, T.; Grimme, S. *Phys. Chem. Chem. Phys.*, **2006**, 8, 4398.

[261] Karton, A.; Tarnopolsky, A; Lamère, J.-F.; Schatz, G. C.; Martin, J. M. L. Highly Accurate First-Principles Benchmark Data Sets for the Parametrization and Validation of Density Functional and Other Approximate Methods. Derivation of a Robust, Generally Applicable, Double-Hybrid Functional for Thermochemistry and Thermochemical Kinetics. *J. Phys. Chem. A*, **2008**, 112, 12868. DOI: 10.1021/jp801805p.

[262] Tarnopolsky, Alex; Karton, Amir; Sertchook, Rotem; Vuzman, Dana; Martin, Jan M L. Double-Hybrid Functionals for Thermochemical Kinetics. *J. Phys. Chem. A*, **2008**, 112 (1), 3–8. DOI: 10.1021/jp710179r.

[263] Yu, Feng. Double-Hybrid Density Functionals Free of Dispersion and Counterpoise Corrections for Non-Covalent Interactions. *J. Phys. Chem. A*, **2014**, 118 (17), 3175–3182. DOI: 10.1021/jp5005506.

[264] Brémond, Éric; Sancho-García, Juan Carlos; Pérez-Jiménez, Ángel José; Adamo, Carlo. *J. Chem. Phys.*, **2014**, 141, 031101.

[265] Brémond, Éric; Adamo, Carlo. Seeking for Parameter-Free Double-Hybrid Functionals: The PBE0-DH Model. *J. Chem. Phys.*, **2011**, 135 (2), 024106.

[266] Casanova-Páez, Marcos; Goerigk, Lars. Time-Dependent Long-Range-Corrected Double-Hybrid Density Functionals with Spin-Component and Spin-Opposite Scaling: A Comprehensive Analysis of Singlet-Singlet and Singlet-Triplet Excitation Energies. *J. Chem. Theory Comput.*, **2021**, 17 (8), 5165–5186. DOI: 10.1021/acs.jctc.1c00535.

[267] Chai, Jeng-Da; Head-Gordon, Martin. Long-Range Corrected Double-Hybrid Density Functionals. *J. Chem. Phys.*, **2009**, 131 (17), 174105. DOI: 10.1063/1.3244209.

[268] Brémond, Éric; Savarese, Marika; Pérez-Jiménez, Ángel José; Sancho-García, Juan Carlos; Adamo, Carlo. *J. Chem. Theory Comput.*, **2018**, 14, 4052–4062.

[269] Brémond, Éric; Pérez-Jiménez, Ángel José; Sancho-García, Juan Carlos; Adamo, Carlo. *J. Chem. Phys.*, **2019**, 150, 201102.

[270] Casanova-Páez, Marcos; Dardis, Michael B.; Goerigk, Lars. ωB2PLYP & ωB2GPPLYP: The First Two Double-Hybrid Density Functionals with Long-Range Correction Optimized for Excitation Energies. *J. Chem. Theory Comput.*, **2019**, 15, 4735. DOI: 10.1021/acs.jctc.9b00013.

[271] Lehtola, S.; Steigemann, C.; Oliveira, MJT; Marques, MAL. Recent Developments in Libxc – A Comprehensive Library of Functionals for Density Functional Theory. *SoftwareX*, **2019**, 7, 1–5. DOI: 10.1016/j.softx.2017.11.002.

[272] Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. *J. Chem. Phys.*, **2010**, 132, 154104.

[273] Grimme, S.; Ehrlich, S.; Goerigk, L. *J. Comput. Chem.*, **2011**, 32, 1456.

[274] Caldeweyher, Eike; Bannwarth, Christoph; Grimme, Stefan. Extension of the D3 Dispersion Coefficient Model. *J. Chem. Phys.*, **2017**, 147 (3), 034112. DOI: 10.1063/1.4993215.

[275] Vydrov, O. A.; Van Voorhis, T. *J. Chem. Phys.*, **2010**, 133, 244103.

[276] Hujo, Waldemar; Grimme, Stefan. Performance of Dispersion-Corrected Density Functional Theory for Thermochemistry and Noncovalent Interactions. *J. Chem. Theory Comput.*, **2011**, 7 (12), 3866–3871. DOI: 10.1021/ct200644w.

[277] Becke, A. D.; Johnson, E. R. A density-functional model of the dispersion interaction. *J. Chem. Phys.*, **2005**, 123, 154101. DOI: 10.1063/1.2065267.

[278] Johnson, Erin R.; Becke, Axel D. A Post-Hartree–Fock Model of Intermolecular Interactions. *J. Chem. Phys.*, **2005**, 123, 024101. DOI: 10.1063/1.1949201.

[279] Johnson, Erin R.; Becke, Axel D. A Post-Hartree–Fock Model of Intermolecular Interactions: Inclusion of Higher-Order Corrections. *J. Chem. Phys.*, **2006**, 124, 174104. DOI: 10.1063/1.2190220.

[280] Grimme, S. *Chem. Eur. J.*, **2012**, 18, 9955–9964.

[281] Grimme, S. Accurate description of van der Waals complexes by density functional theory including empirical corrections. *J. Comput. Chem.*, **2004**, 25, 1463. DOI: 10.1002/jcc.20078.

[282] Lee, K.; Murray, É. D.; Kong, L.; Lundqvist, B. I.; Langreth, D. C. Higher-accuracy van der Waals density functional. *Phys. Rev. B*, **2010**, 82 (8), 081101. DOI: 10.1103/PhysRevB.82.081101.

[283] Goerigk, L.; Grimme, S. *J. Chem. Theory Comput.*, **2010**, 6, 107.

[284] Goerigk, L.; Grimme, S. *Phys. Chem. Chem. Phys.*, **2011**, 13, 6670.

[285] Řezáč, Jan; Riley, Kevin E.; Hobza, Pavel. S66: A Well-balanced Database of Benchmark Interaction Energies Relevant to Biomolecular Structures. *J. Chem. Theory Comput.*, **2011**, 7, 2427–2438. DOI: 10.1021/ct2002946.

[286] Hujo, W.; Grimme, S. *Phys. Chem. Chem. Phys.*, **2011**, 13, 13942.

[287] Iron, Mark A.; Janes, Trevor. Evaluating Transition Metal Barrier Heights with the Latest Density Functional Theory Exchange–Correlation Functionals: The MOBH35 Benchmark Database. *J. Phys. Chem. A*, **2019**, 123 (17), 3761–3781. DOI: 10.1021/acs.jpca.9b01546.

[288] Jurečka, Petr; Šponer, Jiří; Černý, Jiří; Hobza, Pavel. Benchmark Database of Accurate (MP2 and CCSD(T) Complete Basis Set Limit) Interaction Energies of Small Model Complexes, DNA Base Pairs, and Amino Acid Pairs. *Phys. Chem. Chem. Phys.*, **2006**, 8, 1985–1993. DOI: 10.1039/B600027D.

[289] Arago, Juan; Orti, Enrique; Sancho-Garcia, Juan C. Nonlocal van Der Waals Approach Merged with Double-Hybrid Density Functionals: Toward the Accurate Treatment of Noncovalent Interactions. *J. Chem. Theory Comput.*, **2013**, 9 (8), 3437–3443. DOI: 10.1021/ct4003527.

[290] Yu, Feng. Spin-Component-Scaled Double-Hybrid Density Functionals with Nonlocal van Der Waals Correlations for Noncovalent Interactions. *J. Chem. Theory Comput.*, **2014**, 10 (10), 4400–4407. DOI: 10.1021/ct500642x.

[291] Pople, J. A.; Beveridge, D. L. *Approximate Molecular Orbital Theory*. McGraw Hill Inc, **1970**.

[292] Sedlej, J.; Cooper, I. L. *Semi-Emipirical Methods of Quantum Chemistry*. 1985, **John Wiley and Sons**.

[293] Dewar, M. J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. P. *J. Am. Chem. Soc.*, **1985**, 107, 3902.

[294] Stewart, J. P. *J. Comput. Chem.*, **1989**, 10, 209 & 221.

[295] Dewar, M. J. S.; Thiel, W. *Theor. Chim. Acta*, **1977**, 46, 89.

[296] Thiel, W.; Voityuk, A. A. *Theor. Chim. Acta*, **1992**, 81, 391.

[297] Dewar, M. J. S.; Thiel, W. *J. Am. Chem. Soc.*, **1977**, 99, 4899.

[298] Pople, J. A.; Segal, G. A. *J. Chem. Phys.*, **1965**, 43, 136.

[299] Pople, J. A.; Segal, G. A. *J. Chem. Phys.*, **1966**, 44, 3289.

[300] Santry, D. P. *J. Am. Chem. Soc.*, **1968**, 90, 3309.

[301] Santry, D. P.; Segal, G. A. *J. Chem. Phys.*, **1967**, 47, 158.

[302] Pople, J. A.; Beveridge, D. L.; Dobosh, P. A. *J. Chem. Phys.*, **1967**, 47, 2026.

[303] Clack, D. W.; Hush, N. S.; Yandle, J. R. All\textendashValence Electron CNDO Calculations on Transition Metal Complexes. *J. Chem. Phys.*, **1972**, 57, 3503. DOI: 10.1063/1.1678785.

[304] Clack, D. W. INDO MO calculations for first row transition metal complexes. *Mol. Phys.*, **1974**, 27, 1513–1519. DOI: 10.1080/00268977400101281.

[305] Clack, D. W.; Smith, W. Clack, D.W., Smith, W. Molecular orbital calculations on transition metal complexes. *Theor. Chim. Acta*, **1974**, 36, 87–92. DOI: 10.1007/BF00554339.

[306] Böhm, M. C.; Gleiter, R. A CNDO/INDO molecular orbital formalism for the elements H to Br. theory. *Theor. Chim. Acta*, **1981**, 59, 127 & 153. DOI: 10.1007/BF00552536.

[307] Ridley, J.; Zerner, M. C. *Theor. Chim. Acta*, **1973**, 32, 111.

[308] Bacon, A. D.; Zerner, M. C. An intermediate neglect of differential overlap theory for transition metal complexes: Fe, Co and Cu chlorides. *Theor. Chim. Acta*, **1979**, 53, 21. DOI: 10.1007/BF00547605.

[309] Zerner, M. C.; Loew, G. H.; Kirchner, R. F.; Mueller-Westerhoff, U. T. *J. Am. Chem. Soc.*, **1980**, 102, 589.

[310] Anderson, W. P.; Edwards, W. D.; Zerner, M. C. Calculated spectra of hydrated ions of the first transition-metal series. *Inorg. Chem.*, **1986**, 25, 2728. DOI: 10.1021/ic00236a015.

[311] Anderson, W. P.; Cundari, T. R.; Drago, R. S.; Zerner, M. C. Utility of the semiempirical INDO/1 method for the calculation of the geometries of second-row transition-metal species. *Inorg. Chem.*, **1990**, 29, 3. DOI: 10.1021/ic00326a001.

[312] Anderson, W. P.; Cundari, T. R.; Zerner, M. C. An intermediate neglect of differential overlap model for second-row transition metal species. *Int. J. Quant. Chem.*, **1991**, 39, 31. DOI: 10.1002/qua.560390106.

[313] Zerner, M. C. In Lipkowitz, K. B.; Boyd, D. B., editors, *Reviews in Computational Chemistry*, volume 2, pages 313. Wiley-VCH, **1990**.

[314] Zerner, M. C. In Salahub, D. R.; Russo, N., editors, *Metal-Ligand Interactions: From Atoms to Clusters to Surfaces*, pages 101. Kluwer Academic Publishers, **1992**.

[315] Zerner, M. C. In Salahub, D. R.; Russo, N., editors, *Metal-Ligand Interactions: Structure and Reactivity*, pages 493. Kluwer Academic Publishers, **1992**.

[316] Cory, M. G.; Zerner, M. C. Metal-ligand exchange coupling in transition-metal complexes. *Chem. Rev.*, **1991**, 91, 813. DOI: 10.1021/cr00005a009.

[317] Kotzian, M.; Rösch, N.; Zerner, M. C. Intermediate neglect of differential overlap spectroscopic studies on lanthanide complexes. I. Spectroscopic parametrization and application to diatomic lanthanide oxides LnO (Ln=La, Ce, Gd, and Lu). *Theor. Chim. Acta*, **1992**, 81 (3-4), 201–222. DOI: 10.1007/BF01118562.

[318] Nieke, C.; Reinhold, J. *Theor. Chim. Acta*, **1984**, 65, 99.

[319] Köhler, H. J.; Birnstock, F. Title not provided. *Z. Chem.*, **1972**, 12 (5), 196.

[320] Grimme, Stefan; Bannwarth, Christoph; Shushkov, Philip. A Robust and Accurate Tight-Binding Quantum Chemical Method for Structures, Vibrational Frequencies, and Noncovalent Interactions of Large Molecular Systems Parametrized for All Spd-Block Elements (z = 1-86). *J. Chem. Theory Comput.*, **2017**, 13 (5), 1989–2009. DOI: 10.1021/acs.jctc.7b00118.

[321] Pracht, Philipp; Caldeweyher, Eike; Ehlert, Sebastian; Grimme, Stefan. A Robust Non-Self-Consistent Tight-Binding Quantum Chemistry Method for Large Molecules. ChemRxiv, **2019**. DOI: 10.26434/chemrxiv.8326202.v1.

[322] Ehlert, Sebastian; Stahn, Marcel; Spicher, Sebastian; Grimme, Stefan. Robust and Efficient Implicit Solvation Model for Fast Semiempirical Methods. *J. Chem. Theory Comput.*, **2021**, 17 (7), 4250–4261. DOI: 10.1021/acs.jctc.1c00471.

[323] Cancès, E.; Maday, Y.; Stamm, B. Domain decomposition for implicit solvation models. *J. Chem. Phys*, **2013**, 139, 054111. DOI: 10.1063/1.4816767.

[324] Neugebauer, Hagen; Bädorf, Benedikt; Ehlert, Sebastian; Hansen, Andreas; Grimme, Stefan. High-throughput screening of spin states for transition metal complexes with spin-polarized extended tight-binding methods. *J. Comput. Chem.*, **2023**, 44 (27), 2120–2129. DOI: https://doi.org/10.1002/jcc.27185.

[325] Sure, R.; Grimme, S. *J. Comput. Chem.*, **2013**, 34, 1672–1685.

[326] Brandenburg, J. G.; Bannwarth, C.; Hansen, A.; Grimme, S. B97-3c: A Revised Low-Cost Variant of the B97-D Density Functional Method. *J. Chem. Phys.*, **2018**, 148 (6), 064104.

[327] Ehlert, Sebastian; Huniar, Uwe; Ning, Jinliang; Furness, James W.; Sun, Jianwei; Kaplan, Aaron D.; Perdew, John P.; Brandenburg, Jan Gerit. r^2SCAN-D4: Dispersion Corrected Meta-Generalized Gradient Approximation for General Chemical Applications. *J. Chem. Phys.*, **2021**, 154 (6), 061101. arXiv:10.1063/5.0041008, DOI: 10.1063/5.0041008.

[328] Langreth, D.C.; Perdew, J.P. The exchange-correlation energy of a metallic surface. *Solid State Communications*, **1975**, 17 (11), 1425–1429. DOI: 10.1016/0038-1098(75)90618-3.

[329] Furche, Filipp. Molecular tests of the random phase approximation to the exchange-correlation energy functional. *Phys. Rev. B*, **2001**, 64, 195120. DOI: 10.1103/PhysRevB.64.195120.

[330] Furche, Filipp. Developing the random phase approximation into a practical post-Kohn–Sham correlation model. *The Journal of Chemical Physics*, **2008**, 129 (11), 114105. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.2977789/13579843/114105_1_online.pdf, DOI: 10.1063/1.2977789.

[331] Trushin, Egor; Thierbach, Adrian; Görling, Andreas. Toward chemical accuracy at low computational cost: Density-functional theory with σ-functionals for the correlation energy. *The Journal of Chemical Physics*, **2021**, 154 (1), 014104. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0026849/15583499/014104_1_online.pdf, DOI: 10.1063/5.0026849.

[332] Lochan, Rohini C.; Head-Gordon, Martin. Orbital-Optimized Opposite-Spin Scaled Second-Order Correlation: An Economical Method to Improve the Description of Open-Shell Molecules. *J. Chem. Phys.*, **2007**, 126 (16), 164101. DOI: 10.1063/1.2718952.

[333] McWeeny, R. *Methods of Molecular Quantum Mechanics. 2nd Edition*. Academic Press, **1992**.

[334] Cremer, D. In Schleyer, P. v. R., editor, *Encyclopedia of Computational Chemistry*, pages 1706. John Wiley and Sons, **1998**.

[335] Saebo, S.; Almlöf, J. *Chem. Phys. Lett.*, **1989**, 154, 83.

[336] Head-Gordon, M.; Pople, J. A. *Chem. Phys. Lett.*, **1988**, 153, 503.

[337] Lauderdale, W. J.; Stanton, J. F.; Gauss, J.; Watts, J. D.; Bartlett, R. J. Many-body perturbation theory with a restricted open-shell Hartree–Fock reference. *Chem. Phys. Lett.*, **1991**, 187 (1-2), 21–28. DOI: 10.1016/0009-2614(91)90478-R.

[338] Knowles, Peter J.; Andrews, James S.; Amos, Roger D.; Handy, Nicholas C.; Pople, John A. Restricted Møller–Plesset Theory for Open-Shell Molecules. *Chem. Phys. Lett.*, **1991**, 186 (2-3), 130–136. DOI: 10.1016/0009-2614(91)85118-G.

[339] Pople, J. A.; Binkley, J. S.; Seeger, R. *Int. J. Quant. Chem. Symp.*, **1976**, 10, 1.

[340] Krishnan, R.; Frisch, M. J.; Pople, J. A. Contribution of triple substitutions to the electron correlation energy in fourth order perturbation theory. *J. Chem. Phys.*, **1980**, 72 (7), 4244–4245. DOI: 10.1063/1.439657.

[341] Handy, N. C.; Knowles, P. J.; Somasundram, K. *Theor. Chem. Acc.*, **1985**, 68, 87.

[342] Weigend, F.; Häser, M.; Patzelt, H.; Ahlrichs, R. *Chem. Phys. Lett.*, **1998**, 294, 143.

[343] Weigend, F.; Häser, M. *Theor. Chem. Acc.*, **1997**, 97, 331.

[344] Feyereisen, M.; Fitzerald, G.; Komornicki, A. *Chem. Phys. Lett.*, **1993**, 208, 359.

[345] Bernholdt, D. E.; Harrison, R. J. Large-scale correlated electronic structure calculations: the RI-MP2 method on parallel computers. *Chem. Phys. Lett.*, **1996**, 250, 477. DOI: 10.1016/0009-2614(96)00054-1.

[346] Grimme, S. *J. Chem. Phys.*, **2003**, 118, 9095–9102.

[347] Stoychev, Georgi L.; Auer, Alexander A.; Neese, Frank. Efficient and Accurate Prediction of Nuclear Magnetic Resonance Shielding Tensors with Double-Hybrid Density Functional Theory. *J. Chem. Theory Comput.*, **2018**, 14 (9), 4756–4771. DOI: 10.1021/acs.jctc.8b00624.

[348] Tran, Van Anh; Neese, Frank. Double-Hybrid Density Functional Theory for g-Tensor Calculations Using Gauge Including Atomic Orbitals. *J. Chem. Phys.*, **2020**, 153 (5), 054105. DOI: 10.1063/5.0013799.

[349] Pinski, P.; Riplinger, C.; Valeev, E. F.; Neese, Frank. *J. Chem. Phys.*, **2015**, 143, 034108.

[350] Pavošević, F.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E.F. SparseMaps – A systematic infrastructure for reduced-scaling electronic structure methods. IV. Linear-scaling second-order explicitly correlated energy with pair natural orbitals. *J. Chem. Phys.*, **2016**, 144, 144109.

[351] Pinski, Peter; Neese, Frank. Communication: Exact Analytical Derivatives for the Domain-Based Local Pair Natural Orbital MP2 Method (DLPNO-MP2). *J. Chem. Phys.*, **2018**, 148, 031101. DOI: 10.1063/1.5011204.

[352] Pinski, Peter; Neese, Frank. Analytical Gradient for the Domain-Based Local Pair Natural Orbital Second Order Møller-Plesset Perturbation Theory Method (DLPNO-MP2). *J. Chem. Phys.*, **2019**, 150, 164102.

[353] Scheurer, P.; Schwarz, W. H. E. Continuous Degeneracy of Sets of Localized Orbitals. *Int. J. Quantum Chem.*, **2000**, 76, 428–433.

[354] Stoychev, Georgi L.; Auer, Alexander A.; Gauss, Jürgen; Neese, Frank. DLPNO-MP2 Second Derivatives for the Computation of Polarizabilities and NMR Shieldings. *J. Chem. Phys.*, **2021**, 154 (16), 164110. DOI: 10.1063/5.0047125.

[355] Sparta, Manuel; Retegan, Marius; Pinski, Peter; Riplinger, Christoph; Becker, Ute; Neese, Frank. Multilevel Approaches within the Local Pair Natural Orbital Framework. *J. Chem. Theory Comput.*, **2017**, 13 (7), 3198–3207. DOI: 10.1021/acs.jctc.7b00260.

[356] Jensen, F. *Introduction to Computational Chemistry*. Wiley, **1999**.

[357] Koch, W.; Holthausen, M. C. *A Chemist's Guide to Density Functional Theory*. Wiley-VCH, **2000**.

[358] Neese, F.; Schwabe, T.; Kossmann, S.; Schirmer, B.; Grimme, S. *J. Chem. Theory Comput.*, **2009**, 5, 3060.

[359] Shee, James; Loipersberger, Matthias; Rettig, Adam; Lee, Joonho; Head-Gordon, Martin. Regularized Second-Order Møller–Plesset Theory: A More Accurate Alternative to Conventional MP2 for Noncovalent Interactions and Transition Metal Thermochemistry for the Same Computational Cost. *J. Phys. Chem. Lett.*, **2021**, 12 (50), 12084–12097. DOI: 10.1021/acs.jpclett.1c03468.

[360] Neese, F. Importance of Direct Spin-Spin Coupling and Spin-Flip Excitations for the Zero-Field Splittings of Transition Metal Complexes: A Case Study. *J. Am. Chem. Soc.*, **2006**, 128, 10213.

[361] Krupička, Martin; Sivalingam, Kantharuban; Huntington, Lee; Auer, Alexander A.; Neese, Frank. A toolchain for the automatic generation of computer codes for correlated wavefunction calculations. *J. Comput. Chem.*, **2017**, 38 (21), 1853–1868. DOI: 10.1002/jcc.24833.

[362] Lechner, M. H.; Papadopoulos, A.; Sivalingam, K.; Auer, A. A.; Koslowski, A.; Becker, U.; Wennmohs, F.; Neese, F. Code generation in ORCA: Progress, Efficiency and Tight integration. *Phys. Chem. Chem. Phys.*, **2024**, 26 (21), 15205–15220.

[363] Parr, R. G. *Density Functional Theory of Atoms and Molecules*. International Series of Monographs on Chemistry. Oxford University Press, **1994**. ISBN 978-0-19-509276-9.

[364] Ahlrichs, R. Many body perturbation calculations and coupled electron pair models. *Comp. Phys. Comm.*, **1979**, 17, 31. DOI: 10.1016/0010-4655(79)90067-5.

[365] Gdanitz, R. J. *Int. J. Quant. Chem.*, **2001**, 85, 281.

[366] Gdanitz, R. J.; Ahlrichs, R. *Chem. Phys. Lett.*, **0143**, 1988, 413.

[367] Szalay, P. G.; Bartlett, R. J. *Chem. Phys. Lett.*, **1993**, 214, 481.

[368] Ahlrichs, R.; Scharf, P.; Ehrhardt, C. The coupled pair functional (CPF). A size consistent modification of the CI(SD) based on an energy functional. *J. Chem. Phys.*, **1985**, 82, 890. DOI: 10.1063/1.448517.

[369] Kollmar, Christian; Neese, Frank. The coupled electron pair approximation: Variational formulation and spin adaptation. *Mol. Phys.*, **2010**, 108 (19-20), 2449–2458. DOI: 10.1080/00268976.2010.496743.

[370] Kollmar, Christian; Neese, Frank. The relationship between double excitation amplitudes and Z vector components in some post-Hartree-Fock correlation methods. *J. Chem. Phys.*, **2011**, 135 (6), 064103. DOI: 10.1063/1.3618720.

[371] Chong, D. P.; Langhoff, S. R. A modified coupled pair functional approach. *J. Chem. Phys.*, **1986**, 84, 5606–5610. DOI: 10.1063/1.449920.

[372] Scuseria, G. E.; III, H. F. Schaefer. *Chem. Phys. Lett.*, **1987**, 142, 354.

[373] Handy, N. C.; Pople, J. A.; Head-Gordon, M.; Raghavachari, K.; Trucks, G. W. *Chem. Phys. Lett.*, **1989**, 164, 185.

[374] Kollmar, Christian; Hesselmann, Andreas. The role of orbital transformations in coupled-pair functionals. *Theor. Chem. Acc.*, **2010**, 127 (5-6), 311–320. DOI: 10.1007/s00214-010-0846-z.

[375] Salter, E. A.; Trucks, G. W.; Bartlett, R. J. *J. Chem. Phys.*, **1989**, 90, 1752.

[376] Kollmar, Christian; Neese, Frank. An orbital-invariant and strictly size extensive post-Hartree-Fock correlation functional. *J. Chem. Phys.*, **2011**, 135 (8), 084102. DOI: 10.1063/1.3624567.

[377] Huntington, L. M. J.; Nooijen, M. *J. Chem. Phys.*, **2010**, 133, 184109.

[378] Huntington, Lee M. J.; Hansen, Andreas; Neese, Frank; Nooijen, Marcel. Accurate Thermochemistry from a Parameterized Coupled-Cluster Singles and Doubles Model and a Local Pair Natural Orbital Based Implementation for Applications to Larger Systems. *J. Chem. Phys.*, **2012**, 136, 064101. DOI: 10.1063/1.3682325.

[379] Pulay, P.; Saebo, S.; Meyer, W. *J. Chem. Phys.*, **1984**, 81, 1901.

[380] Hampel, C.; Peterson, K. A.; Werner, H. J. *Chem. Phys. Lett.*, **1992**, 190, 1.

[381] Scuseria, G. E.; Janssen, C. L.; III, H. F. Schaefer. *J. Chem. Phys.*, **1988**, 89, 7382.

[382] Heully, J. L.; Malrieu, J.-P. *J. Mol. Struct.: THEOCHEM*, **2006**, 768, 53.

[383] Neese, F.; Hansen, A.; Liakos, D. G. *J. Chem. Phys.*, **2009**, 131, 064103.

[384] Hansen, A.; Liakos, D. G.; Neese, F. *J. Chem. Phys.*, **2011**, 135, 214102.

[385] Neese, F.; Hansen, A.; Wennmohs, F.; Grimme, S. *Acc. Chem. Res.*, **2009**, 42, 641.

[386] Neese, F.; Liakos, D. G.; Ye, S. F. *J. Biol. Inorg. Chem.*, **2011**, 16, 821.

[387] Semidalas, Emmanouil; Martin, Jan M. L. Automatic generation of complementary auxiliary basis sets for explicitly correlated methods. *J. Comput. Chem.*, **2022**, 43 (25), 1690–1700. DOI: 10.1002/jcc.26970.

[388] Liakos, Dimitrios G.; Izsák, Róbert; Valeev, Edward F.; Neese, Frank. What is the most efficient way to reach the canonical MP2 basis set limit? *Mol. Phys.*, **2013**, 111 (19-20), 2653–2662. DOI: 10.1080/00268976.2013.811812.

[389] Liakos, D. G.; Neese, F. Improved correlation energy extrapolation schemes based on local pair natural orbital methods. *J. Phys. Chem. A*, **2012**, 116 (19), 4801–4816. DOI: 10.1021/jp300997x.

[390] Li, S.; Ma, J.; Jiang, Y. Linear Scaling Local Correlation Approach for Solving the Coupled Cluster Equations of Large Systems. *J. Comput. Chem.*, **2002**, 23, 237–244. DOI: 10.1002/jcc.10003.

[391] Li, S.; Shen, J.; Li, W.; Jiang, Y. An Efficient Implementation of the "Cluster-in-Molecule" Approach for Local Electron Correlation Calculations. *J. Chem. Phys.*, **2006**, 125, 074109. DOI: 10.1063/1.2244566.

[392] Li, W.; Piecuch, P.; Gour, J.; Li, S. Local Correlation Calculations Using Standard and Renormalized Coupled-Cluster Approaches. *J. Chem. Phys.*, **2009**, 131, 114109. DOI: 10.1063/1.3218842.

[393] Rolik, Z.; Kallay, M. A General-Order Local Coupled-Cluster Method Based on the Cluster-in-Molecule Approach. *J. Chem. Phys.*, **2011**, 135, 104111. DOI: 10.1063/1.3632085.

[394] Guo, Y.; Li, W.; Li, S. Improved Cluster-in-Molecule Local Correlation Approach for Electron Correlation Calculation of Large Systems. *J. Phys. Chem. A*, **2014**, 118 (39), 8996–9004. DOI: 10.1021/jp501976x.

[395] Guo, Yang; Becker, Ute; Neese, Frank. Comparison and Combination of "Direct" and Fragment Based Local Correlation Methods: Cluster in Molecules and Domain Based Local Pair Natural Orbital Perturbation and Coupled Cluster Theories. *J. Chem. Phys.*, **2018**, 148 (12), 124117. DOI: 10.1063/1.5021898.

[396] Förner, W.; Ladik, J.; Otto, P.; Čížek, J. Coupled-Cluster Studies. II. The Role of Localization in Correlation Calculations on Extended Systems. *Chem. Phys.*, **1985**, 97, 251–262. DOI: 10.1016/0301-0104(85)87035-X.

[397] Riplinger, C.; Neese, F. *J. Chem. Phys.*, **2013**, 138, 034106.

[398] Neese, F.; Wennmohs, F.; Hansen, A. *J. Chem. Phys.*, **2009**, 130, 114108.

[399] Liakos, Dimitrios G.; Hansen, Andreas; Neese, Frank. Weak molecular interactions studied with parallel implementations of the local pair natural orbital coupled pair and coupled cluster methods. *J. Chem. Theory Comput.*, **2011**, 7 (1), 76–87. DOI: 10.1021/ct100529u.

[400] Riplinger, C.; Sandhoefer, B.; Hansen, A.; Neese, F. *J. Chem. Phys.*, **2013**, 139, 134101.

[401] Riplinger, C.; Pinski, P.; Becker, U.; Valeev, E. F.; Neese, Frank. *J. Chem. Phys.*, **2016**, 144, 024109.

[402] Datta, D.; Kossmann, S.; Neese, F. Analytic energy derivatives for the calculation of the first-order molecular properties using the domain-based local pair-natural orbital coupled-cluster theory. *J. Chem. Phys.*, **2016**, 175, 114101. DOI: 10.1063/1.4962369.

[403] Saitow, M.; Becker, U.; Riplinger, C.; Valeev, E. F.; Neese, F. *J. Chem. Phys.*, **2017**, 146, 164105.

[404] Guo, Yang; Riplinger, Christoph; Becker, Ute; Liakos, Dimitrios G.; Minenkov, Yury; Cavallo, Luigi; Neese, Frank. Communication: An Improved Linear Scaling Perturbative Triples Correction for the Domain Based Local Pair-Natural Orbital Based Singles and Doubles Coupled Cluster Method [DLPNO-CCSD(T)]. *J. Chem. Phys.*, **2018**, 148 (1), 011101. DOI: 10.1063/1.5011798.

[405] Schütz, M.; Werner, H. J. *Chem. Phys. Lett.*, **2000**, 318, 370.

[406] Schütz, M.; Werner, H. J. *J. Chem. Phys.*, **2001**, 114, 661.

[407] Schütz, Martin; Manby, Frederick R. Linear scaling local coupled cluster theory with density fitting. Part I: 4-external integrals. *Phys. Chem. Chem. Phys.*, **2003**, 5 (16), 3349–3358.

[408] Schütz, Martin. A new, fast, semi-direct implementation of linear scaling local coupled cluster theory. *Phys. Chem. Chem. Phys.*, **2002**, 4 (16), 3941–3947.

[409] Meyer, Wilfried. Ionization energies of water from PNO-CI calculations. *Int. J. Quantum Chem.*, **1971**, 5 (S5), 341–348.

[410] Ahlrichs, R; Lischka, H; Staemmler, V; Kutzelnigg, W. PNO–CI (pair natural orbital configuration interaction) and CEPA–PNO (coupled electron pair approximation with pair natural orbitals) calculations of molecular systems. I. Outline of the method for closed-shell states. *J. Chem. Phys.*, **1975**, 62 (4), 1225–1234.

[411] Meyer, Wilfried. PNO–CI Studies of electron correlation effects. I. Configuration expansion by means of nonorthogonal orbitals, and application to the ground state and ionized states of methane. *J. Chem. Phys.*, **1973**, 58 (3), 1017–1035.

[412] Werner, Hans-Joachim; Meyer, Wilfried. PNO-CI and PNO-CEPA studies of electron correlation effects: V. Static dipole polarizabilities of small molecules. *Mol. Phys.*, **1976**, 31 (3), 855–872.

[413] Liakos, Dimitrios G.; Sparta, Manuel; Kesharwani, Manoj K.; Martin, Jan M. L.; Neese, Frank. Exploring the Accuracy Limits of Local Pair Natural Orbital Coupled-Cluster Theory. *J. Chem. Theory Comput.*, **2015**, 11 (4), 1525–1539. DOI: 10.1021/acs.jctc.5b00078.

[414] Altun, Ahmet; Riplinger, Christoph; Neese, Frank; Bistoni, Giovanni. Exploring the Accuracy Limits of PNO-Based Local Coupled-Cluster Calculations for Transition-Metal Complexes. *J. Chem. Theory Comput.*, **2023**, 19 (7), 2039–2047.

[415] Lee, T. J.; Taylor, P. R. A diagnostic for determining the quality of single-reference electron correlation methods. *Int. J. Quant. Chem. Symp.*, **1989**, 23, 199–207. DOI: 10.1002/qua.560360824.

[416] Wennmohs, F.; Neese, F. *Chem. Phys.*, **2008**, 343, 217–230. DOI: .

[417] Ahlrichs, R.; Scharf, P. The Coupled Pair Approximation. In Lawley, K. P., editor, *Advances in Chemical Physics: Ab Initio Methods in Quantum Chemistry, Part I*, Advances in Chemical Physics, pages 1–42. Wiley, **1987**.

[418] Siegbahn, Per E. M. Direct Configuration Interaction with a Reference State Composed of Many Reference Configurations. *Int. J. Quant. Chem.*, **1980**, 18 (5), 1229–1242. DOI: 10.1002/qua.560180510.

[419] Meyer, Wilfried. Configuration Expansion by Means of Pseudonatural Orbitals. In Schaefer III, Henry F., editor, *Methods of Electronic Structure Theory*, pages 413–446. Springer US, **1977**.

[420] Sivalingam, K.; Krupicka, M.; Auer, A. A.; Neese, F. Comparison of fully internally and strongly contracted multireference configuration interaction procedures. *J. Chem. Phys.*, **2016**, 145, 054104. DOI: 10.1063/1.4950161.

[421] Saitow, Masaaki; Kurashige, Yuki; Yanai, Takeshi. Multireference Configuration Interaction Theory Using Cumulant Reconstruction with Internal Contraction of Density Matrix Renormalization Group Wave Function. *J. Chem. Phys.*, **2013**, 139, 044118. DOI: 10.1063/1.4816627.

[422] Guo, Yang; Sivalingam, Kantharuban; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). I. Revisiting the NEVPT2 Construction. *J. Chem. Phys.*, **2021**, 154 (21), 214111. DOI: 10.1063/5.0051211.

[423] Lyakh, Dmitry I.; Musiał, Monika; Lotrich, Victor F.; Bartlett, Rodney J. Multireference Nature of Chemistry: The Coupled-Cluster View. *Chem. Rev.*, **2012**, 112 (1), 182–243. DOI: 10.1021/cr2001417.

[424] Evangelista, Francesco A.; Gauss, Jürgen. An Orbital-Invariant Internally Contracted Multireference Coupled Cluster Approach. *J. Chem. Phys.*, **2011**, 134 (11), 114102. DOI: 10.1063/1.3559149.

[425] Hanauer, Matthias; Köhn, Andreas. Pilot Applications of Internally Contracted Multireference Coupled Cluster Theory, and How to Choose the Cluster Operator Properly. *J. Chem. Phys.*, **2011**, 134 (20), 204111. DOI: 10.1063/1.3592786.

[426] Jankowski, K.; Paldus, J. Applicability of Coupled-Pair Theories to Quasidegenerate Electronic States: A Model Study. *Int. J. Quantum Chem.*, **1980**, 18 (5), 1243–1269. DOI: 10.1002/qua.560180511.

[427] Guo, Yang; Sivalingam, Kantharuban; Kollmar, Christian; Neese, Frank. Approximations of Density Matrices in N-Electron Valence State Second-Order Perturbation Theory (NEVPT2). II. The Full Rank NEVPT2 (FR-NEVPT2) Formulation. *J. Chem. Phys.*, **2021**, 154 (21), 214113. DOI: 10.1063/5.0051218.

[428] Angeli, Celestino; Bories, Benoît; Cavallini, Alex; Cimiraglia, Renzo. Third-order multireference perturbation theory: The n-electron valence state perturbation-theory approach. *J. Chem. Phys.*, **2006**, 124 (5), 054108. DOI: 10.1063/1.2148946.

[429] Kempfer, Emily M.; Sivalingam, Kantharuban; Neese, Frank. Efficient Implementation of Approximate Fourth Order N-Electron Valence State Perturbation Theory. *J. Chem. Theory Comput.*, **2025**, 21 (8), 3953–3967. DOI: 10.1021/acs.jctc.4c01735.

[430] Kollmar, Christian; Sivalingam, Kantharuban; Helmich-Paris, Benjamin; Angeli, Celestino; Neese, Frank. A perturbation-based super-CI approach for the orbital optimization of a CASSCF wave function. *J. Comput. Chem.*, **2019**, 40 (14), 1463–1470. DOI: 10.1002/jcc.25801.

[431] Helmich-Paris, Benjamin. A trust-region augmented Hessian implementation for state-specific and state-averaged CASSCF wave functions. *J. Chem. Phys.*, **2022**, 156 (20), 204104. Publisher: American Institute of Physics. DOI: 10.1063/5.0090447.

[432] Lang, Lucas; Neese, Frank. Spin-Dependent Properties in the Framework of the Dynamic Correlation Dressed Complete Active Space Method. *J. Chem. Phys.*, **2019**, 150 (10), 104104. DOI: 10.1063/1.5085203.

[433] Atanasov, Mihail; Ganyushin, Dmitry; Sivalingam, Kantharuban; Neese, Frank. A Modern First-Principles View on Ligand Field Theory Through the Eyes of Correlated Multireference Wavefunctions. In Mingos, David Michael P.; Day, Peter; Dahl, Jens Peder, editors, *Molecular Electronic Structures of Transition Metal Complexes II*, number 143 in Structure and Bonding, pages 149–220. Springer Berlin Heidelberg, **2011**. DOI: 10.1007/430_2011_57.

[434] Lang, Lucas; Atanasov, Mihail; Neese, Frank. Improvement of Ab Initio Ligand Field Theory by Means of Multistate Perturbation Theory. *J. Phys. Chem. A*, **2020**, 124 (5), 1025–1037. DOI: 10.1021/acs.jpca.9b11227.

[435] Atanasov, Mihail; Ganyushin, Dmitry; Pantazis, Dimitrios A.; Sivalingam, Kantharuban; Neese, Frank. Detailed Ab Initio First-Principles Study of the Magnetic Anisotropy in a Family of Trigonal Pyramidal Iron(II) Pyrrolide Complexes. *Inorg. Chem.*, **2011**, 50 (16), 7460–7477. DOI: 10.1021/ic200196k.

[436] Suturina, Elizaveta A.; Maganas, Dimitrios; Bill, Eckhard; Atanasov, Mihail; Neese, Frank. Magneto-Structural Correlations in a Series of Pseudotetrahedral [CoII(XR)4]2– Single Molecule Magnets: An Ab Initio Ligand Field Study. *Inorg. Chem.*, **2015**, 54 (20), 9948–9961. DOI: 10.1021/acs.inorgchem.5b01706.

[437] Aravena, Daniel; Atanasov, Mihail; Neese, Frank. Periodic Trends in Lanthanide Compounds through the Eyes of Multireference Ab Initio Theory. *Inorg. Chem.*, **2016**, 55 (9), 4457–4469. DOI: 10.1021/acs.inorgchem.6b00244.

[438] Jung, Julie; Atanasov, Mihail; Neese, Frank. Ab Initio Ligand-Field Theory Analysis and Covalency Trends in Actinide and Lanthanide Free Ions and Octahedral Complexes. *Inorg. Chem.*, **2017**, 56 (15), 8802–8816. DOI: 10.1021/acs.inorgchem.7b00642.

[439] Singh, Saurabh Kumar; Eng, Julien; Atanasov, Mihail; Neese, Frank. Covalency and Chemical Bonding in Transition Metal Complexes: An Ab Initio Based Ligand Field Perspective. *Coordin. Chem. Rev.*, **2017**, 344, 2–25. DOI: 10.1016/j.ccr.2017.03.018.

[440] Chakraborty, Uttam; Demeshko, Serhiy; Meyer, Franc; Rebreyend, Christophe; de Bruin, Bas; Atanasov, Mihail; Neese, Frank; Mühldorf, Bernd; Wolf, Robert. Electronic Structure and Magnetic Anisotropy of an Unsaturated Cyclopentadienyl Iron(I) Complex with 15 Valence Electrons. *Angew. Chem. Int. Ed.*, **2017**, 56 (27), 7995–7999. DOI: 10.1002/anie.201702454.

[441] Chilkuri, Vijay Gopal; DeBeer, Serena; Neese, Frank. Revisiting the Electronic Structure of FeS Monomers Using Ab Initio Ligand Field Theory and the Angular Overlap Model. *Inorg. Chem.*, **2017**, 56 (17), 10418–10436. DOI: 10.1021/acs.inorgchem.7b01371.

[442] Chilkuri, Vijay Gopal; DeBeer, Serena; Neese, Frank. Ligand Field Theory and Angular Overlap Model Based Analysis of the Electronic Structure of Homovalent Iron–Sulfur Dimers. *Inorg. Chem.*, **2020**, 59 (2), 984–995. DOI: 10.1021/acs.inorgchem.9b00974.

[443] Chilkuri, Vijay Gopal; Neese, Frank. Comparison of Many-Particle Representations for Selected-CI I: A Tree Based Approach. *J. Comput. Chem.*, **2021**, 42 (14), 982–1005. DOI: 10.1002/jcc.26518.

[444] Chilkuri, Vijay Gopal; Neese, Frank. Comparison of Many-Particle Representations for Selected Configuration Interaction: II. Numerical Benchmark Calculations. *J. Chem. Theory Comput.*, **2021**, 17 (5), 2868–2885. DOI: 10.1021/acs.jctc.1c00081.

[445] Chan, G. K.-L.; Head-Gordon, M. Highly correlated calculations with a polynomial cost algorithm: A study of the density matrix renormalization group. *J. Chem. Phys.*, **2002**, 116, 4462–4476. DOI: 10.1063/1.1449459.

[446] Chan, G. K.-L. An algorithm for large scale density matrix renormalization group calculations. *J. Chem. Phys.*, **2004**, 120, 3172. DOI: 10.1063/1.1638734.

[447] Ghosh, D.; Hachmann, J.; Yanai, T.; Chan, G. K.-L. *J. Chem. Phys.*, **2008**, 128, 144117.

[448] Sharma, S.; Chan, G. K.-L. *J. Chem. Phys.*, **2012**, 136, 124121.

[449] Sayfutyarova, Elvira R.; Sun, Qiming; Chan, Garnet Kin-Lic; Knizia, Gerald. Automated Construction of Molecular Active Spaces from Atomic Valence Orbitals. *J. Chem. Theory Comput.*, **2017**, 13 (9), 4063–4078. DOI: 10.1021/acs.jctc.7b00128.

[450] Sayfutyarova, Elvira R.; Hammes-Schiffer, Sharon. Constructing Molecular π-Orbital Active Spaces for Multireference Calculations of Conjugated Systems. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1679–1689. DOI: 10.1021/acs.jctc.8b01196.

[451] Robin, Melvin B. *Higher Excited States of Polyatomic Molecules*. Academic Press, **1974**. ISBN 978-0-12-589901-7.

[452] Walzl, K. N.; Koerting, C. F.; Kuppermann, A. Electron-impact Spectroscopy of Acetaldehyde. *J. Chem. Phys.*, **1987**, 87, 3796–3803. DOI: 10.1063/1.452935.

[453] Müller, Thomas; Lischka, Hans. Simultaneous Calculation of Rydberg and Valence Excited States of Formaldehyde. *Theor. Chem. Acc.*, **2001**, 106, 369–378. DOI: 10.1007/s002140100264.

[454] Lewin, Mathieu. *J. Math. Chem.*, **2008**, 44, 967.

[455] Lang, Lucas. *Development of New Multistate Multireference Perturbation Theory Methods and Their Application*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, **2020**.

[456] Rao, Shashank V.; Maganas, Dimitrios; Sivalingam, Kantharuban; Atanasov, Mihail; Neese, Frank. Extended Active Space Ab Initio Ligand Field Theory: Applications to Transition-Metal Ions. *Inorg. Chem.*, **2024**, 63 (52), 24672–24684. Publisher: American Chemical Society. DOI: 10.1021/acs.inorgchem.4c03893.

[457] Chatzis, A.; Kowalska, J. K.; Maganas, D.; DeBeer, S.; Neese, F. Ab Initio Wave Function-Based Determination of Element Specific Shifts for the Efficient Calculation of X-Ray Absorption Spectra of Main Group Elements and First Row Transition Metals. *J. Chem. Theory Comput.*, **2018**, 14 (7), 3686–3702. DOI: https://dx.doi.org/10.1021/acs.jctc.8b00249.

[458] Mathe, Zachary; Maganas, Dimitrios; Neese, Frank; DeBeer, Serena. Coupling experiment and theory to push the state-of-the-art in x-ray spectroscopy. *Nature Reviews Chemistry*, **2025**, 2397–3358. DOI: 10.1038/s41570-025-00718-2.

[459] Guo, Yang; Sivalingam, Kantharuban; Chilkuri, Vijay Gopal; Neese, Frank. Approximations of density matrices in N-electron valence state second-order perturbation theory (NEVPT2). III. Large active space calculations with selected configuration interaction reference. *J. Chem. Phys.*, **2025**, 162 (14), 144110. DOI: 10.1063/5.0262473.

[460] Chan, G. K.-L.; Sharma, S. *Ann. Rev. Phys. Chem.*, **2011**, 62, 465.

[461] Chan, G. K.-L. DMRG Homepage. URL: https://www.chan-lab.caltech.edu/software.

[462] Fiedler, M. *Czech. Math. J.*, **1973**, 23, 298.

[463] Fiedler, M. *Czech. Math. J.*, **1975**, 25, 619.

[464] Atkins, J. E.; Boman, E. G.; Hendrickson, B. A Spectral Algorithm for Seriation and the Consecutive Ones Problem. *SIAM J. Computing.*, **1998**, 28 (1), 297–310. DOI: 10.1137/S0097539795285771.

[465] Barcza, G.; Legeza, Ö.; Marti, K. H.; Reiher, M. Quantum-information analysis of electronic states of different molecular structures. *Phys. Rev. A*, **2011**, 83, 012508. DOI: 10.1103/PhysRevA.83.012508.

[466] Angeli, C.; Cimiraglia, R.; Evangelisti, S.; Leininger, T.; Malrieu, J.P. Introduction of n-electron valence states for multireference perturbation theory. *J. Chem. Phys.*, **2001**, 114, 10252–10264. DOI: 10.1063/1.1361246.

[467] Angeli, C.; Cimiraglia, R.; Malrieu, J.P. N-electron valence state perturbation theory: a fast implementation of the strongly contracted variant. *Chem. Phys. Lett.*, **2001**, 350, 297–305. DOI: 10.1016/S0009-2614(01)01303-3.

[468] Angeli, C.; Cimiraglia, R.; Malrieu, J.P. n-electron valence state perturbation theory: A spinless formulation and an efficient implementation of the strongly contracted and of the partially contracted variants. *J. Chem. Phys.*, **2002**, 117, 9138–9153. DOI: 10.1063/1.1515317.

[469] Dyall, K. G. *J. Chem. Phys.*, **1995**, 102, 4909–4918.

[470] Havenith, Remco W. A.; Taylor, Peter R.; Angeli, Celestino; Cimiraglia, Renzo; Ruud, Kenneth. Calibration of the N-Electron Valence State Perturbation Theory Approach. *J. Chem. Phys.*, **2004**, 120, 4619. DOI: 10.1063/1.1645243.

[471] Schapiro, Igor; Sivalingam, Kantharuban; Neese, Frank. Assessment of N-Electron Valence State Perturbation Theory for Vertical Excitation Energies. *J. Chem. Theory Comput.*, **2013**, 9 (8), 3567–3580. DOI: 10.1021/ct400136y.

[472] Angeli, Celestino; Borini, Stefano; Cestari, Mirko; Cimiraglia, Renzo. A Quasidegenerate Formulation of the Second Order N-Electron Valence State Perturbation Theory Approach. *J. Chem. Phys.*, **2004**, 121, 4043–4049. DOI: 10.1063/1.1778711.

[473] Lang, Lucas; Sivalingam, Kantharuban; Neese, Frank. The Combination of Multipartitioning of the Hamiltonian with Canonical Van Vleck Perturbation Theory Leads to a Hermitian Variant of Quasidegenerate N-Electron Valence Perturbation Theory. *J. Chem. Phys.*, **2020**, 152 (1), 014109. DOI: 10.1063/1.5133746.

[474] Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. SparseMaps—A Systematic Infrastructure for Reduced-Scaling Electronic Structure Methods. III. Linear-Scaling Multireference Domain-Based Pair Natural Orbital N-Electron Valence Perturbation Theory. *J. Chem. Phys.*, **2016**, 144 (9), 094111. DOI: 10.1063/1.4942769.

[475] Guo, Yang; Sivalingam, Kantharuban; Valeev, Edward F.; Neese, Frank. Explicitly Correlated N-Electron Valence State Perturbation Theory (NEVPT2-F12). *J. Chem. Phys.*, **2017**, 147 (6), 064110. DOI: 10.1063/1.4996560.

[476] Guo, Yang; Pavošević, Fabijan; Sivalingam, Kantharuban; Becker, Ute; Valeev, Edward F.; Neese, Frank. SparseMaps—A systematic infrastructure for reduced-scaling electronic structure methods. VI. Linear-scaling explicitly correlated N-electron valence state perturbation theory with pair natural orbital. *J. Chem. Phys.*, **2023**, 158 (12), 124120. DOI: 10.1063/5.0144260.

[477] Kong, Liguo; Valeev, Edward F. Perturbative correction for the basis set incompleteness error of complete-active-space self-consistent field. *The Journal of Chemical Physics*, **2010**, 133 (17), 174126. DOI: 10.1063/1.3499600.

[478] Dyall, Kenneth G. The choice of a zeroth-order Hamiltonian for second-order perturbation theory with a complete active space self-consistent-field reference function. *The Journal of Chemical Physics*, **1995**, 102 (12), 4909–4918. DOI: 10.1063/1.469539.

[479] Kollmar, Christian; Sivalingam, Kantharuban; Guo, Yang; Neese, Frank. An efficient implementation of the NEVPT2 and CASPT2 methods avoiding higher-order density matrices. *J. Chem. Phys.*, **2021**, 155 (23), 234104. DOI: 10.1063/5.0072129.

[480] Chatterjee, Koushik; Sokolov, Alexander Yu. Extended Second-Order Multireference Algebraic Diagrammatic Construction Theory for Charged Excitations. *J. Chem. Theory Comput.*, **2020**, 16 (10), 6343–6357. DOI: 10.1021/acs.jctc.0c00778.

[481] Zgid, Dominika; Ghosh, Debashree; Neuscamman, Eric; Chan, Garnet Kin-Lic. A Study of Cumulant Approximations to N-Electron Valence Multireference Perturbation Theory. *J. Chem. Phys.*, **2009**, 130, 194107.

[482] Forsberg, Niclas; Malmqvist, Per-Åke. Multiconfiguration perturbation theory with imaginary level shift. *Chem. Phys. Lett.*, **1997**, 274 (1–3), 196–204. DOI: 10.1016/S0009-2614(97)00669-6.

[483] Guo, Sheng; Watson, Mark A.; Hu, Weifeng; Sun, Qiming; Chan, Garnet Kin-Lic. N-Electron Valence State Perturbation Theory Based on a Density Matrix Renormalization Group Reference Function, with Applications

to the Chromium Dimer and a Trimer Model of Poly(p-Phenylenevinylene). *J. Chem. Theory Comput.*, **2016**, 12 (4), 1583–1591. DOI: 10.1021/acs.jctc.5b01225.

[484] Khedkar, Abhishek; Roemelt, Michael. Active Space Selection Based on Natural Orbital Occupation Numbers from N-Electron Valence Perturbation Theory. *J. Chem. Theory Comput.*, **2019**, 15, 3522–3536.

[485] Domingo, A.; Carvajal, M.-A.; de Graaf, C.; Sivalingam, K.; Neese, F.; Angeli, C. *Theor. Chem. Acc.*, **2012**, 131 (9), 1264.

[486] Angeli, C.; Borini, S.; Cestari, M.; Cimigraglia, R. A quasidegenerate formulation of the second order n-electron valence state perturbation theory approach. *J. Chem. Phys.*, **2004**, 121, 4043. DOI: 10.1063/1.1778711.

[487] Shavir, I.; Redmon, L. T. *J. Chem. Phys.*, **1980**, 73, 5711.

[488] Brandow, B.H.". *Effective Interactions and Operators in Nuclei*. Volume 40. Springer, **1974**.

[489] Andersson, Kerstin.; Malmqvist, Per Aake.; Roos, Bjoern O.; Sadlej, Andrzej J.; Wolinski, Krzysztof. Second-Order Perturbation Theory with a CASSCF Reference Function. *J. Phys. Chem.*, **1990**, 94, 5483–5488. DOI: 10.1021/j100377a012.

[490] Roos, Björn O.; Andersson, Kerstin. Multiconfigurational Perturbation Theory with Level Shift — the Cr2 Potential Revisited. *Chem. Phys. Lett.*, **1995**, 245, 215–223. DOI: 10.1016/0009-2614(95)01010-7.

[491] Forsberg, Niclas; Malmqvist, Per-Å ke. Multiconfiguration Perturbation Theory with Imaginary Level Shift. *Chem. Phys. Lett.*, **1997**, 274, 196–204. DOI: 10.1016/S0009-2614(97)00669-6.

[492] Fdez. Galván, Ignacio; Vacher, Morgane; Alavi, Ali; Angeli, Celestino; Aquilante, Francesco; Autschbach, Jochen; Bao, Jie J.; Bokarev, Sergey I.; Bogdanov, Nikolay A.; Carlson, Rebecca K.; Chibotaru, Liviu F.; Creutzberg, Joel; Dattani, Nike; Delcey, Mickaël G.; Dong, Sijia S.; Dreuw, Andreas; Freitag, Leon; Frutos, Luis Manuel; Gagliardi, Laura; Gendron, Frédéric; Giussani, Angelo; González, Leticia; Grell, Gilbert; Guo, Meiyuan; Hoyer, Chad E.; Johansson, Marcus; Keller, Sebastian; Knecht, Stefan; Kovačević, Goran; Källman, Erik; Manni, Giovanni Li; Lundberg, Marcus; Ma, Yingjin; Mai, Sebastian; Malhado, João Pedro; Malmqvist, Per Åke; Marquetand, Philipp; Mewes, Stefanie A.; Norell, Jesper; Olivucci, Massimo; Oppel, Markus; Phung, Quan Manh; Pierloot, Kristine; Plasser, Felix; Reiher, Markus; Sand, Andrew M.; Schapiro, Igor; Sharma, Prachi; Stein, Christopher J.; Sørensen, Lasse Kragh; Truhlar, Donald G.; Ugandi, Mihkel; Ungur, Liviu; Valentini, Alessio; Vancoillie, Steven; Veryazov, Valera; Weser, Oskar; Wesołowski, Tomasz A.; Widmark, Per-Olof; Wouters, Sebastian; Zech, Alexander; Zobel, J. Patrick; Lindh, Roland. OpenMolcas: From Source Code to Insight. *J. Chem. Theory Comput.*, **2019**, 15 (11), 5925–5964.

[493] Kepenekian, Mikaël; Robert, Vincent; Le Guennic, Boris. What Zeroth-Order Hamiltonian for CASPT2 Adiabatic Energetics of Fe(II)N6 Architectures? *J. Chem. Phys.*, **2009**, 131, 114702. DOI: 10.1063/1.3211020.

[494] Zobel, J. Patrick; Nogueira, Juan J.; González, Leticia. The IPEA Dilemma in CASPT2. *Chem. Sci.*, **2017**, 8 (2), 1482–1499. DOI: 10.1039/C6SC03759C.

[495] Kollmar, Christian; Sivalingam, Kantharuban; Neese, Frank. An Alternative Choice of the Zeroth-Order Hamiltonian in CASPT2 Theory. *J. Chem. Phys.*, **2020**, 152 (21), 214110. DOI: 10.1063/5.0010019.

[496] Shiozaki, Toru; Győrffy, Werner; Celani, Paolo; Werner, Hans-Joachim. Communication: Extended Multi-State Complete Active Space Second-Order Perturbation Theory: Energy and Nuclear Gradients. *J. Chem. Phys.*, **2011**, 135, 081106–081106–4.

[497] Li Manni, Giovanni; Carlson, Rebecca K.; Luo, Sijie; Ma, Dongxia; Olsen, Jeppe; Truhlar, Donald G.; Gagliardi, Laura. Multiconfiguration Pair-Density Functional Theory. *J. Chem. Theory Comput.*, **2014**, 10 (9), 3669–3680. PMID: 26588512. DOI: 10.1021/ct500483t.

[498] Becke, A. D.; Savin, A.; Stoll, H. Extension of the local-spin-density exchange-correlation approximation to multiplet states. *Theor. Chim. Acta*, **1995**, 91 (3), 147–156. DOI: 10.1007/BF01114982.

[499] Rodrigues, Gabriel L. S.; Scott, Mikael; Delcey, Mickael G. Multiconfigurational Pair-Density Functional Theory Is More Complex than You May Think. *J. Phys. Chem. A*, **2023**, 127 (44), 9381–9388. DOI: 10.1021/acs.jpca.3c05663.

[500] Scott, Mikael; Rodrigues, Gabriel L. S.; Li, Xin; Delcey, Mickael G. Variational Pair-Density Functional Theory: Dealing with Strong Correlation at the Protein Scale. *J. Chem. Theory Comput.*, **2024**, 20 (6), 2423–2432. PMID: 38217859. DOI: 10.1021/acs.jctc.3c01240.

[501] Fromager, Emmanuel; Toulouse, Julien; Jensen, Hans Jørgen Aa. On the universality of the long-/short-range separation in multiconfigurational density-functional theory. *J. Chem. Phys.*, **2007**, 126 (7), 074111. DOI: 10.1063/1.2566459.

[502] Hedegård, Erik Donovan; Toulouse, Julien; Jensen, Hans Jørgen Aagaard. Multiconfigurational short-range density-functional theory for open-shell systems. *J. Chem. Phys.*, **2018**, 148 (21), 214103. DOI: 10.1063/1.5013306.

[503] Paziani, Simone; Moroni, Saverio; Gori-Giorgi, Paola; Bachelet, Giovanni B. Local-spin-density functional for multideterminant density functional theory. *Phys. Rev. B*, **2006**, 73, 155111. DOI: 10.1103/PhysRevB.73.155111.

[504] Goll, Erich; Werner, Hans-Joachim; Stoll, Hermann. A short-range gradient-corrected density functional in long-range coupled-cluster calculations for rare gas dimers. *Phys. Chem. Chem. Phys.*, **2005**, 7, 3917–3923. DOI: 10.1039/B509242F.

[505] Goll, Erich; Werner, Hans-Joachim; Stoll, Hermann; Leininger, Thierry; Gori-Giorgi, Paola; Savin, Andreas. A short-range gradient-corrected spin density functional in combination with long-range coupled-cluster methods: Application to alkali-metal rare-gas dimers. *Chem. Phys.*, **2006**, 329 (1), 276 – 282. DOI: https://doi.org/10.1016/j.chemphys.2006.05.020.

[506] Helmich-Paris, Benjamin; Kjellgren, Erik Rosendahl; Jensen, Hans Jørgen Aa. Excited-State Methods Based on State-Averaged Long-Range CASSCF Short-Range DFT. under review in Phys. Chem. Chem. Phys., **2025**.

[507] Pedersen, Jesper Kielberg. *Description of correlation and relativistic effects in calculations of molecular properties*. PhD thesis, University of Southern Denmark, **2004**.

[508] Neese, Frank; Petrenko, Taras; Ganyushin, Dmitry; Olbrich, Gottfried. Advanced Aspects of Ab Initio Theoretical Optical Spectroscopy of Transition Metal Complexes: Multiplets, Spin-Orbit Coupling and Resonance Raman Intensities. *Coordin. Chem. Rev.*, **2007**, 251 (3-4), 288–327. DOI: 10.1016/j.ccr.2006.05.019.

[509] Retegan, Marius; Cox, Nicholas; Pantazis, Dimitrios A.; Neese, Frank. A First-Principles Approach to the Calculation of the on-Site Zero-Field Splitting in Polynuclear Transition Metal Complexes. *Inorg. Chem.*, **2014**, 53 (21), 11785–11793. DOI: 10.1021/ic502081c.

[510] Neese, F. *Chem. Phys. Lett.*, **2003**, 380, 721–728.

[511] Maganas, Dimitrios; Sottini, Silvia; Kyritsis, Panayotis; Groenen, Edgar J. J.; Neese, Frank. Theoretical Analysis of the Spin Hamiltonian Parameters in Co(II)S_4 Complexes, Using Density Functional Theory and Correlated ab initio Methods. *Inorg. Chem.*, **2011**, 50 (18), 8741–8754. DOI: 10.1021/ic200299y.

[512] Jiang, Shang-Da; Maganas, Dimitrios; Levesanos, Nikolaos; Ferentinos, Eleftherios; Haas, Sabrina; Thirunavukkuarasu, Komalavalli; Krzystek, Jurek; Dressel, Martin; Bogani, Lapo; Neese, Frank; Kyritsis, Panayotis. Direct Observation of Very Large Zero-Field Splitting in a Tetrahedral Ni(II)Se_4 Coordination Complex. *J. Am. Chem. Soc.*, **2015**, 137 (40), 12923–12928. DOI: 10.1021/jacs.5b06716.

[513] Ganyushin, D.; Neese, F. *J. Chem. Phys.*, **2008**, 128, 114117.

[514] Maganas, Dimitrios; Kowalska, Joanna K.; Nooijen, Marcel; DeBeer, Serena; Neese, Frank. Comparison of Multireference Ab Initio Wavefunction Methodologies for X-Ray Absorption Edges: A Case Study on $[Fe(II/III)Cl_4]^{2-}/^{1-}$ Molecules. *J. Chem. Phys.*, **2019**, 150 (10), 104106. DOI: 10.1063/1.5051613.

[515] Pollock, Christopher J.; Delgado-Jaime, Mario Ulises; Atanasov, Mihail; Neese, Frank; DeBeer, Serena. Kβ Mainline X-Ray Emission Spectroscopy as an Experimental Probe of Metal–Ligand Covalency. *J. Am. Chem. Soc.*, **2014**, 136 (26), 9453–9463. DOI: 10.1021/ja504182n.

[516] Datta, Dipayan; Kong, Liguo; Nooijen, Marcel. A State-Specific Partially Internally Contracted Multireference Coupled Cluster Approach. *J. Chem. Phys.*, **2011**, 134 (21), 214116. DOI: 10.1063/1.3592494.

[517] Datta, Dipayan; Nooijen, Marcel. Multireference Equation-of-Motion Coupled Cluster Theory. *J. Chem. Phys.*, **2012**, 137 (20), 204107. DOI: 10.1063/1.4766361.

[518] Demel, Ondrej; Datta, Dipayan; Nooijen, Marcel. Additional Global Internal Contraction in Variations of Multireference Equation of Motion Coupled Cluster Theory. *J. Chem. Phys.*, **2013**, 138 (13), 134108.

[519] Nooijen, Marcel; Demel, Ondrej; Datta, Dipayan; Kong, Liguo; Shamasundar, K. R.; Lotrich, V.; Huntington, Lee M.; Neese, Frank. Communication: Multireference Equation of Motion Coupled Cluster: A Transform and Diagonalize Approach to Electronic Structure. *J. Chem. Phys.*, **2014**, 140 (8), 081102.

[520] Huntington, L. M. J.; Nooijen, M. *J. Chem. Phys.*, **2015**, 142, 194111.

[521] Huntington, L. M. J.; Nooijen, M. *J. Chem. Theory Comput.*, **2016**, 12, 114.

[522] Liu, Z.; Demel, O.; Nooijen, M. Multireference Equation of Motion Coupled Cluster Study of Atomic Excitation Spectra of First-Row Transition Metal Atoms Cr, Mn, Fe and Co. *J. Mol. Spectrosc.*, **2015**, 311, 54.

[523] Liu, Z.; Huntington, L. M. J.; Nooijen, M. Application of the multireference equation of motion coupled cluster method, including spin–orbit coupling, to the atomic spectra of Cr, Mn, Fe and Co. *Mol. Phys.*, **2015**, 113 (19-20), 2999–3013. DOI: 10.1080/00268976.2015.1063730.

[524] Lechner, Marvin H.; Izsák, Róbert; Nooijen, Marcel; Neese, Frank. A Perturbative Approach to Multireference Equation-of-Motion Coupled Cluster. *Mol. Phys.*, **2021**, 119 (17-18), e1939185. DOI: 10.1080/00268976.2021.1939185.

[525] Mukherjee, D. Normal Ordering and a Wick-like Reduction Theorem for Fermions with Respect to a Multi-Determinantal Reference State. *Chem. Phys. Lett.*, **1997**, 274, 561–568. DOI: 10.1016/S0009-2614(97)00714-8.

[526] Kutzelnigg, Werner; Mukherjee, Debashis. Normal order and extended Wick theorem for a multiconfiguration reference wave function. *J. Chem. Phys.*, **1997**, 107 (2), 432–449. DOI: 10.1063/1.474405.

[527] National Institute of Standards and Technology (NIST) Atomic Spectra Database.

[528] Nave, G.; Johansson, S.; Learner, R. C. M.; Thorne, A. P.; Brault, J. W. *Astrophys. J., Suppl. Ser.*, **1994**, 94, 221.

[529] Pathak, Shubhrodeep; Lang, Lucas; Neese, Frank. A Dynamic Correlation Dressed Complete Active Space Method: Theory, Implementation, and Preliminary Applications. *J. Chem. Phys.*, **2017**, 147, 234109.

[530] Maurice, Rémi; Bastardis, Roland; de Graaf, Coen; Suaud, Nicolas; Mallah, Talal; Guihéry, Nathalie. Universal Theoretical Approach to Extract Anisotropic Spin Hamiltonians. *J. Chem. Theory Comput.*, **2009**, 5 (11), 2977–2984. DOI: 10.1021/ct900365q.

[531] Li, H.; Jensen, J. H. Partial Hessian vibrational analysis: The localization of the molecular vibrational energy and entropy. *Theor. Chem. Acc.*, **2002**, 107 (4), 211–219. DOI: 10.1007/s00214-001-0317-7.

[532] Steinbach, Peter J.; Brooks, Bernard R. New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation. *J. Comput. Chem.*, **1994**, 15 (7), 667–683. DOI: 10.1002/jcc.540150702.

[533] Schlegel, H. B. In Lawley, K. P., editor, *Advances in Chemical Physics: Ab Initio Methods in Quantum Chemistry, Part I*, volume 67, pages 249. John Wiley and Sons, **1987**.

[534] Schlegel, H. B. In Yarkony, D. R., editor, *Modern Electronic Structure Theory*, pages 459. World Scientific, **1995**.

[535] Schlegel, H. B. In Schleyer, P. v. R., editor, *Encyclopedia of Computational Chemistry*, pages 1136. John Wiley and Sons, **1998**.

[536] Eckert, F.; Pulay, P.; Werner, H. J. *J. Comput. Chem.*, **1997**, 12, 1473.

[537] Horn, H.; Wei, H.; Häser, M.; Ehrig, M.; Ahlrichs, R. *J. Comput. Chem.*, **1991**, 12, 1058.

[538] Baker, J. An algorithm for the location of transition states. *J. Comput. Chem.*, **1986**, 7, 385. DOI: 10.1002/jcc.540070402.

[539] Ribas-Arino, Jordi; Marx, Dominik. Covalent Mechanochemistry: Theoretical Concepts and Computational Tools with Applications to Molecular Nanomechanics. *Chem. Rev.*, **2012**, 112 (10), 5412–5487. DOI: 10.1021/cr200399q.

[540] Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. *J. Chem. Theory Comput.*, **2008**, 4, 435.

[541] Harvey, J. N.; Aschi, M.; Schwarz, H.; Koch, W. *Theor. Chem. Acc.*, **1998**, 99, 95.

[542] Ishida, Kazuhiro; Morokuma, Keiji; Komornicki, Andrew. The Intrinsic Reaction Coordinate. An Ab Initio Calculation for HNC\textrightarrow HCN and H-+CH4\textrightarrow CH4+H-. *J. Chem. Phys.*, **1977**, 66 (5), 2153–2156. DOI: 10.1063/1.434152.

[543] Ásgeirsson, V.; B.O., Birgirsson; Bjornsson, R.; Becker, U.; Riplinger, C.; Neese, F.; Jónssson, H. Nudged Elastic Band Method for Molecular Reactions Using Energy-Weighted Springs Combined with Eigenvector Following. *J. Chem. Theory Comput.*, **2021**, 17, 4929. DOI: 10.1021/acs.jctc.1c00462.

[544] Mills, G.; Jónsson, H.; Schenter, G. K. Reversible Work Transition State Theory: Application to Dissociative Adsorption of Hydrogen. *Surf. Sci.*, **1995**, 324 (2-3), 305–337. DOI: 10.1016/0039-6028(94)00731-4.

[545] Jónsson, H.; Mills, G.; Jacobsen, K.W. *Classical and Quantum Dynamics in Condensed Phase Simulations*. World Scientific Publishing Company, **1998**.

[546] Henkelman, G.; Jónsson, H. Improved Tangent Estimate in the Nudged Elastic Band Method for Finding Minimum Energy Paths and Saddle Points. *J. Chem. Phys.*, **2000**, 113 (22), 9978–9985.

[547] Zhu, Ting; Li, Ju; Samanta, Amit; Kim, Hyoung Gyu; Suresh, Subra. Interfacial Plasticity Governs Strain Rate Sensitivity and Ductility in Nanostructured Metals. *Proc. Nat. Acad. Sci.*, **2007**, 104 (9), 3031–3036. arXiv:https://www.pnas.org/content/104/9/3031.full.pdf, DOI: 10.1073/pnas.0611097104.

[548] Ásgeirsson, Vilhjálmur; Arnaldsson, Andri; Jónsson, Hannes. Efficient Evaluation of Atom Tunneling Combined with Electronic Structure Calculations. *J. Chem. Phys.*, **2018**, 148 (10), 102334. DOI: 10.1063/1.5007180.

[549] Henkelman, G.; Uberuaga, B.P.; Jónsson, H. A Climbing Image Nudged Elastic Band Method for Finding Saddle Points and Minimum Energy Paths. *J. Chem. Phys.*, **2000**, 113 (22), 9901–9904.

[550] Maras, Emile; Trushin, Oleg; Stukowski, Alexander; Ala-Nissila, Tapio; Jónsson, Hannes. Global Transition Path Search for Dislocation Formation in Ge on Si (001). *Comput. Phys. Commun.*, **2016**, 205, 13–21. DOI: 10.1016/j.cpc.2016.04.001.

[551] Sheppard, D.; Terrell, R.; Henkelman, G. Optimization Methods for Finding Minimum Energy Paths. *J. Chem. Phys.*, **2008**, 128 (13), 134106.

[552] Trygubenko, S.A.; Wales, D.J. A Doubly Nudged Elastic Band Method for Finding Transition States. *J. Chem. Phys.*, **2004**, 120 (5), 2082–2094.

[553] Ásgeirsson, V. *Development and Evaluation of Computational Methods for Studies of Chemical Reactions*. University of Iceland, **2021**.

[554] Bitzek, Erik; Koskinen, Pekka; Gähler, Franz; Moseler, Michael; Gumbsch, Peter. Structural Relaxation Made Simple. *Phys. Rev. Lett.*, **2006**, 97 (17), 170201. DOI: 10.1103/PhysRevLett.97.170201.

[555] Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. *Math. Comput.*, **1980**, 35 (151), 773–782. DOI: 10.1090/S0025-5718-1980-0572855-7.

[556] Müller, Klaus; Brown, Leo D. Location of Saddle Points and Minimum Energy Paths by a Constrained Simplex Optimization Procedure. *Theor. Chem. Acc.*, **1979**, 53 (1), 75–93. DOI: 10.1007/BF00547608.

[557] Smidstrup, Søren; Pedersen, Andreas; Stokbro, Kurt; Jónsson, Hannes. Improved Initial Guess for Minimum Energy Path Calculations. *J. Chem. Phys.*, **2014**, 140 (21), 214106. DOI: 10.1063/1.4878664.

[558] Schmerwitz, Yorick Leonard Adrian; Ásgeirsson, Vilhjálmur; Jónsson, Hannes. Improved Initialization of Optimal Path Calculations Using Sequential Traversal over the Image-Dependent Pair Potential Surface. *J. Chem. Theory Comput.*, **2024**, 20, 155–163. DOI: 10.1021/acs.jctc.3c01111.

[559] Melander, Marko; Laasonen, Kari; Jónsson, Hannes. Removing External Degrees of Freedom from Transition-State Search Methods Using Quaternions. *J. Chem. Theory Comput.*, **2015**, 11 (3), 1055–1062. DOI: 10.1021/ct501103z.

[560] Kairys, V.; Head, J.D. Geometry Optimization of Charged Molecules in an External Electric Field Applied to $F^{-}\cdot H_2O$ and $I^{-}\cdot H_2O$. *J. Phys. Chem. A*, **1998**, 102 (8), 1365–1370.

[561] E, Weinan; Ren, Weiqing; Vanden-Eijnden, Eric. String Method for the Study of Rare Events. *Phys. Rev. B*, **2002**, 66 (5), 052301. arXiv:0205527 [cond-mat], DOI: 10.1103/PhysRevB.66.052301.

[562] Shang, Honghui; Yang, Jinlong. The Moving-Grid Effect in the Harmonic Vibrational Frequency Calculations with Numeric Atom-Centered Orbitals. *J. Phys. Chem. A*, **2020**, 124 (14), 2897–2906. Publisher: American Chemical Society. DOI: 10.1021/acs.jpca.0c01453.

[563] Herzberg, G. *Infrared and Raman Spectra*. Van Nostrand Reinhold, **1945**.

[564] Gilson, Michael K.; Irikura, Karl K. Symmetry Numbers for Rigid, Flexible, and Fluxional Molecules: Theory and Applications. *J. Phys. Chem. B*, **2010**, 114 (50), 16304–16317. DOI: 10.1021/jp110434s.

[565] Maeda, Satoshi; Ohno, Koichi; Morokuma, Keiji. Updated Branching Plane for Finding Conical Intersections without Coupling Derivative Vectors. *J. Chem. Theory Comput.*, **2010**, 6 (5), 1538–1545. DOI: 10.1021/ct1000268.

[566] de Souza, Bernardo. GOAT: A Global Optimization Algorithm for Molecules and Atomic Clusters. *Angew. Chem. Int. Ed.*, **2025**, 64 (18), e202500393. DOI: 10.1002/anie.202500393.

[567] Wales, David J.; Doye, Jonathan P. K. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *J. Phys. Chem. A*, **1997**, 101 (28), 5111–5116. Publisher: American Chemical Society. DOI: 10.1021/jp970984n.

[568] Goedecker, Stefan. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *J. Chem. Phys.*, **2004**, 120 (21), 9911–9917. Publisher: American Institute of Physics. DOI: 10.1063/1.1724816.

[569] Pracht, Philipp; Bohle, Fabian; Grimme, Stefan. Automated exploration of the low-energy chemical space with fast quantum chemical methods. *Phys. Chem. Chem. Phys.*, **2020**, 22 (14), 7169–7192. Publisher: The Royal Society of Chemistry. DOI: 10.1039/C9CP06869D.

[570] Pracht, Philipp; Grimme, Stefan. Calculation of absolute molecular entropies and heat capacities made simple. *Chem. Sci.*, **2021**, 12 (19), 6551–6568. Publisher: The Royal Society of Chemistry. DOI: 10.1039/D1SC00621E.

[571] Assadollahzadeh, Behnam; Schwerdtfeger, Peter. A systematic search for minimum structures of small gold clusters Aun (n=2–20) and their electronic properties. *J. Chem. Phys.*, **2009**, 131 (6), 064306. Publisher: American Institute of Physics. DOI: 10.1063/1.3204488.

[572] Spicher, Sebastian; Plett, Christoph; Pracht, Philipp; Hansen, Andreas; Grimme, Stefan. Automated Molecular Cluster Growing for Explicit Solvation by Efficient Force Field and Tight Binding Methods. *J. Chem. Theory Comput.*, **2022**, 18 (5), 3174–3189. Publisher: American Chemical Society. DOI: 10.1021/acs.jctc.2c00239.

[573] Shami, Tareq M.; El-Saleh, Ayman A.; Alswaitti, Mohammed; Al-Tashi, Qasem; Summakieh, Mhd Amen; Mirjalili, Seyedali. Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, **2022**, 10, 10031–10061. DOI: 10.1109/ACCESS.2022.3142859.

[574] Mulliken, R. S. Electronic Population Analysis on LCAO–MO Molecular Wave Functions. I. *J. Chem. Phys.*, **1955**, 23 (10), 1833–1840. DOI: 10.1063/1.1740588.

[575] Wiberg, K. B. *Tetrahedron*, **1968**, 24, 1083. DOI: .

[576] Mayer, István. Charge, Bond Order and Valence in the Ab Initio SCF Theory. *Chem. Phys. Lett.*, **1983**, 97 (3), 270–274. DOI: 10.1016/0009-2614(83)80005-0.

[577] Mayer, I. *Int. J. Quant. Chem.*, **1984**, 26, 151.

[578] Mayer, I. *Theor. Chim. Acta*, **1985**, 67, 315.

[579] Mayer, I. In Maksić, Z. B., editor, *Modelling of Structure and Properties of Molecules*. John Wiley and Sons, **1987**.

[580] Hirshfeld, F. L. Bonded-Atom Fragments for Describing Molecular Charge Densities. *Theor. Chim. Acta*, **1977**, 44, 129–138.

[581] Verstraelen, Toon; Vandenbrande, Steven; Heidar-Zadeh, Farnaz; Vanduyfhuys, Louis; Van Speybroeck, Veronique; Waroquier, Michel; W. Ayers, Paul. Minimal Basis Iterative Stockholder: Atoms in Molecules for Force-Field Development. *J. Chem. Theory Comput.*, **2016**, 12, 3894–3912. DOI: 10.1021/acs.jctc.6b00456.

[582] Breneman, C. M.; Wiberg, K. B. Determining atom-centered monopoles from molecular electrostatic potentials. The need for high sampling density in formamide conformational analysis. *J. Comput. Chem.*, **1990**, 11, 361–373. DOI: 10.1002/jcc.540110311.

[583] Bayly, Christopher I.; Cieplak, Piotr; Cornell, Wendy; Kollman, Peter A. A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: the RESP model. *The Journal of Physical Chemistry*, **1993**, 97 (40), 10269–10280. Publisher: American Chemical Society. DOI: 10.1021/j100142a004.

[584] Clark, Aurora E.; Davidson, Ernest R. Local Spin. *J. Chem. Phys.*, **2001**, 115 (16), 7382–7392. DOI: 10.1063/1.1407276.

[585] McWeeny, R.; Kutzelnigg, W. Comparison of different independent particle model approximations. *Int. J. Quantum Chem.*, **1968**, 2 (2), 187–203. DOI: 10.1002/qua.560020207.

[586] Herrmann, Carmen; Reiher, Markus; Hess, Bernd A. Comparative Analysis of Local Spin Definitions. *J. Chem. Phys.*, **2005**, 122 (3), 034102. DOI: 10.1063/1.1829050.

[587] Bohmann, Jonathan A.; Weinhold, Frank; Farrar, Thomas C. Natural Chemical Shielding Analysis of Nuclear Magnetic Resonance Shielding Tensors from Gauge-Including Atomic Orbital Calculations. *J. Chem. Phys.*, **1997**, 107 (4), 1173–1184. DOI: 10.1063/1.474464.

[588] Grimme, S.; Hansen, A. A Practicable Real-Space Measure and Visualization of Static Electron-Correlation Effects. *Angew. Chem. Int. Ed.*, **2015**, 54, 12308–12313.

[589] Roemelt, M.; Neese, F. Excited States of Large Open-Shell Molecules: An Efficient, General, and Spin-Adapted Approach Based on a Restricted Open-Shell Ground State Wave function. *J. Phys. Chem. A*, **2013**, 117, 3069–3082. DOI: 10.1021/jp3126126.

[590] Roemelt, Michael; Maganas, Dimitrios; DeBeer, Serena; Neese, Frank. Excited States of Large Open-Shell Molecules: An Efficient, General, and Spin-Adapted Approach Based on a Restricted Open-Shell Ground State Wave function. *The Journal of Chemical Physics*, **2013**, 138, 204101. DOI: 10.1063/1.4804607.

[591] Leyser da Costa Gouveia, Tiago; Maganas, Dimitrios; Neese, Frank. General Spin-Restricted Open-Shell Configuration Interaction Approach: Application to Metal K-Edge X-ray Absorption Spectra of Ferro- and Antiferromagnetically Coupled Dimers. *The Journal of Physical Chemistry A*, **2025**, 129 (1), 330–345. PMID: 39680653. arXiv:https://doi.org/10.1021/acs.jpca.4c05228, DOI: 10.1021/acs.jpca.4c05228.

[592] Yeager, Danny L.; Jørgensen, Poul. A Multiconfigurational Time-Dependent Hartree–Fock Approach. *Chem. Phys. Lett.*, **1979**, 65, 77–80.

[593] Jørgensen, Poul; Jensen, Hans Jørgen Aagaard; Olsen, Jeppe. Linear Response Calculations for Large Scale Multiconfiguration Self-Consistent Field Wave Functions. *J. Chem. Phys.*, **1988**, 89 (6), 3654–3661.

[594] Helmich-Paris, Benjamin. CASSCF Linear Response Calculations for Large Open-Shell Molecules. *J. Chem. Phys.*, **2019**, 150 (17), 174121. DOI: 10.1063/1.5092613.

[595] Martin, Richard L. Natural Transition Orbitals. *J. Chem. Phys.*, **2003**, 118 (11), 4775–4777. DOI: 10.1063/1.1558471.

[596] Helmich-Paris, Benjamin. Benchmarks for Electronically Excited States with CASSCF Methods. *J. Chem. Theory Comput.*, **2019**, 15 (7), 4170–4179. DOI: 10.1021/acs.jctc.9b00325.

[597] Berraud-Pache, Romain; Neese, Frank; Bistoni, Giovanni; Izsák, Róbert. Unveiling the Photophysical Properties of Boron-Dipyrromethene Dyes Using a New Accurate Excited State Coupled Cluster Method. *J. Chem. Theory Comput.*, **2020**, 16 (1), 564–575. DOI: 10.1021/acs.jctc.9b00559.

[598] Sirohiwal, Abhishek; Berraud-Pache, Romain; Neese, Frank; Izsák, Róbert; Pantazis, Dimitrios A. Accurate Computation of the Absorption Spectrum of Chlorophyll a with Pair Natural Orbital Coupled Cluster Methods. *J. Phys. Chem. B*, **2020**, 124 (40), 8761–8771. DOI: 10.1021/acs.jpcb.0c05761.

[599] Dittmer, Anneke; Izsák, Róbert; Neese, Frank; Maganas, Dimitrios. Accurate Band Gap Predictions of Semiconductors in the Framework of the Similarity Transformed Equation of Motion Coupled Cluster Theory. *Inorg. Chem.*, **2019**, 58 (14), 9303–9315. DOI: 10.1021/acs.inorgchem.9b00994.

[600] de Souza, Bernardo; Neese, Frank; Izsak, Robert. On the Theoretical Prediction of Fluorescence Rates from First Principles Using the Path Integral Approach. *J. Chem. Phys.*, **2018**, 148 (3), 034104. DOI: 10.1063/1.5010895.

[601] de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Predicting Phosphorescence Rates of Light Organic Molecules Using Time-Dependent Density Functional Theory and the Path Integral Approach to Dynamics. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1896–1904. DOI: 10.1021/acs.jctc.8b00841.

[602] Santoro, Fabrizio; Improta, Roberto; Lami, Alessandro; Bloino, Julien; Barone, Vincenzo. Effective method to compute Franck-Condon integrals for optical spectra of large molecules in solution. *J. Chem. Phys.*, **2007**, 126 (8), 084509. DOI: 10.1063/1.2437197.

[603] Duschisnky, F. *Acta Physicochim. URSS*, **1937**, 7, 551.

[604] Strickler, S. J.; Berg, Robert A. Relationship between Absorption Intensity and Fluorescence Lifetime of Molecules. *J. Chem. Phys.*, **1962**, 37 (4), 814–822. DOI: 10.1063/1.1733166.

[605] Serpa, Carlos; Arnaut, Luis G.; Formosinho, Sebastião J.; Naqvi, K. Razi. Calculation of Triplet–Triplet Energy Transfer Rates from Emission and Absorption Spectra. The Quenching of Hemicarcerated Triplet Biacetyl by Aromatic Hydrocarbons. *Photochem. Photobiol. Sci.*, **2003**, 2 (5), 616–623. DOI: 10.1039/B300049D.

[606] Mori, K.; Goumans, T. P. M.; van Lenthe, E.; Wang, F. Predicting Phosphorescent Lifetimes and Zero-Field Splitting of Organometallic Complexes with Time-Dependent Density Functional Theory Including Spin–Orbit Coupling. *Phys. Chem. Chem. Phys.*, **2014**, 16 (28), 14523–14530. DOI: 10.1039/C3CP55438D.

[607] Montalti, Marco; Credi, Alberto; Prodi, Luca; Gandolfi, M. Teresa. *Handbook of Photochemistry, Third Edition*. CRC Press, 3 edition edition, 02 **2006**. ISBN 978-0-8247-2377-4.

[608] Hunter, T. F.; Wyatt, R. F. Intersystem Crossing in Anthracene. *Chem. Phys. Lett.*, **1970**, 6 (3), 221–224. DOI: 10.1016/0009-2614(70)80224-X.

[609] de Souza, Bernardo; Farias, Giliandro; Neese, Frank; Izsak, Robert. Efficient Simulation of Overtones and Combination Bands in Resonant Raman Spectra. *J. Chem. Phys.*, **2019**, 150 (21), 044105. DOI: 10.1063/1.5099247.

[610] Long, Derek A. *The Raman Effect: A Unified Treatment of the Theory of Raman Scattering by Molecules*. Wiley, 1 edition edition, 11 **2001**. ISBN 978-0-471-49028-9.

[611] Tripathi, G. N. R.; Schuler, Robert H. The Resonance Raman Spectrum of Phenoxyl Radical. *J. Chem. Phys.*, **1984**, 81 (1), 113–121. DOI: 10.1063/1.447373.

[612] Shafei, Rami; Hamano, Ai; Gourlaouen, Christophe; Maganas, Dimitrios; Takano, Keiko; Daniel, Chantal; Neese, Frank. Theoretical spectroscopy for unraveling the intensity mechanism of the optical and photoluminescent spectra of chiral Re(I) transition metal complexes. *J. Chem. Phys.*, **2023**, 159 (8), 084102.

[613] Hodecker, Manuel; Biczysko, Malgorzata; Dreuw, Andreas; Barone, Vincenzo. Simulation of Vacuum UV Absorption and Electronic Circular Dichroism Spectra of Methyl Oxirane: The Role of Vibrational Effects. *J. Chem. Theory Comput.*, **2016**, 12 (6), 2820–2833. DOI: 10.1021/acs.jctc.6b00121.

[614] Crassous, Jeanne. *Circularly Polarized Luminescence in Helicene and Helicenoid Derivatives*, pages 53–97. Springer Singapore, Singapore, **2020**. DOI: 10.1007/978-981-15-2309-0_4.

[615] Nishimura, Hidetaka; Tanaka, Kazuo; Morisaki, Yasuhiro; Chujo, Yoshiki; Wakamiya, Atsushi; Murata, Yasujiro. Oxygen-Bridged Diphenylnaphthylamine as a Scaffold for Full-Color Circularly Polarized Luminescent Materials. *J. Org. Chem.*, **2017**, 82 (10), 5242–5249. DOI: 10.1021/acs.joc.7b00511.

[616] Foglia, Nicolás; De Souza, Bernardo; Maganas, Dimitrios; Neese, Frank. Including vibrational effects in magnetic circular dichroism spectrum calculations in the framework of excited state dynamics. *J. Chem. Phys.*, **2023**, 158 (15), 154108. DOI: 10.1063/5.0144845.

[617] Cerezo, Javier; Zuniga, José; Requena, Alberto; Avila Ferrer, Francisco J.; Santoro, Fabrizio. Harmonic Models in Cartesian and Internal Coordinates to Simulate the Absorption Spectra of Carotenoids at Finite Temperatures. *J. Chem. Theory Comput.*, **2013**, 9 (11), 4947–4958. DOI: 10.1021/ct4005849.

[618] Jr, E. Bright Wilson; Decius, J. C.; Cross, Paul C. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*. Dover Publications, revised ed. edition edition, 03 **1980**. ISBN 978-0-486-63941-3.

[619] Baker, Jon. Constrained Optimization in Delocalized Internal Coordinates. *J. Comput. Chem.*, **1997**, 18 (8), 1079–1095. DOI: 10.1002/(SICI)1096-987X(199706)18:8<1079::AID-JCC12>3.0.CO;2-8.

[620] Reimers, Jeffrey R. A Practical Method for the Use of Curvilinear Coordinates in Calculations of Normal-Mode-Projected Displacements and Duschinsky Rotation Matrices for Large Molecules. *J. Chem. Phys.*, **2001**, 115 (20), 9103–9109. DOI: 10.1063/1.1412875.

[621] Swart, Marcel; Matthias Bickelhaupt, F. Optimization of Strong and Weak Coordinates. *Int. J. Quantum Chem.*, **2006**, 106 (12), 2536–2544. DOI: 10.1002/qua.21049.

[622] Lindh, Roland; Bernhardsson, Anders; Schütz, Martin. Force-Constant Weighted Redundant Coordinates in Molecular Geometry Optimizations. *Chem. Phys. Lett.*, **1999**, 303 (5), 567–575. DOI: 10.1016/S0009-2614(99)00247-X.

[623] Sando, Gerald M.; Spears, Kenneth G. Ab Initio Computation of the Duschinsky Mixing of Vibrations and Nonlinear Effects. *J. Phys. Chem. A*, **2001**, 105 (22), 5326–5333. DOI: 10.1021/jp004230b.

[624] Dymarsky, Anatoly Y.; Kudin, Konstantin N. Computation of the Pseudorotation Matrix to Satisfy the Eckart Axis Conditions. *J. Chem. Phys.*, **2005**, 122 (12), 124103. DOI: 10.1063/1.1864872.

[625] Petrenko, Taras; Kossmann, Simone; Neese, Frank. Efficient Time-Dependent Density Functional Theory Approximations for Hybrid Density Functionals: Analytical Gradients and Parallelization. *J. Chem. Phys.*, **2011**, 134 (5), 054116. DOI: 10.1063/1.3533441.

[626] Neese, F. *J. Biol. Inorg. Chem.*, **2006**, 11, 702.

[627] Neese, F. *Coord. Chem. Rev.*, **2009**, 253, 526.

[628] Izsák, Róbert; Neese, Frank. An Overlap Fitted Chain of Spheres Exchange Method. *J. Chem. Phys.*, **2011**, 135, 144105. DOI: 10.1063/1.3644029.

[629] Petrenko, T.; Kossmann, S.; Neese, F. *J. Chem. Phys.*, **2011**, 134, 054116.

[630] Casanova, David; Krylov, Anna I. Spin-Flip Methods in Quantum Chemistry. *Phys. Chem. Chem. Phys.*, **2020**, 22 (8), 4326–4342. DOI: 10.1039/C9CP06507E.

[631] Clark, Aurora E.; Davidson, Ernest R. P-Benzyne Derivatives That Have Exceptionally Small Singlet-Triplet Gaps and Even a Triplet Ground State. *J. Org. Chem.*, **2003**, 68 (9), 3387–3396. DOI: 10.1021/jo026824b.

[632] Cammi, Roberto; Mennucci, Benedetta; Tomasi, Jacopo. Fast Evaluation of Geometries and Properties of Excited Molecules in Solution: A Tamm-Dancoff Model with Application to 4-Dimethylaminobenzonitrile. *J. of Phys. Chem. A*, **2000**, 104 (23), 5631–5637. DOI: 10.1021/jp000156l.

[633] Grimme, S. A simplified Tamm–Dancoff density functional approach for the electronic excitation spectra of very large molecules. *J. Chem. Phys.*, **2013**, 138, 244104. DOI: 10.1063/1.4811330.

[634] Bannwarth, C.; Grimme, S. A simplified time-dependent density functional theory approach for electronic ultraviolet and circular dichroism spectra of very large molecules. *Comp. Theor. Chem.*, **2014**, 1040 –1041, 45–53. DOI: 10.1016/j.comptc.2014.02.023.

[635] Risthaus, T.; Hansen, A.; Grimme, S. *Phys. Chem. Chem. Phys.*, **2014**, 16, 14408–14419.

[636] Head-Gordon, M.; Rico, R. A.; Oumi, M.; Lee, T. J. *Chem. Phys. Lett.*, **1994**, 219, 21–29.

[637] Rhee, Young Min; Head-Gordon, Martin. *J. Phys. Chem. A*, **2007**, 111, 5314–5326.

[638] Goerigk, Lars; Grimme, Stefan. *J. Chem. Phys.*, **2010**, 132, 184103.

[639] Casanova-Páez, M.; Goerigk, L. Assessing the Tamm–Dancoff approximation, singlet–singlet, and singlet–triplet excitations with the latest long-range corrected double-hybrid density functionals. *J. Chem. Phys.*, **2020**, 153, 064106.

[640] Schwabe, T.; Goerigk, L. *J. Chem. Theory Comput.*, **2017**, 13, 4307.

[641] Grimme, S.; Neese, F. *J. Phys. Chem.*, **2007**, 127, 154116.

[642] Goerigk, L. Moellmann; Grimme, S. *Phys. Chem. Chem. Phys.*, **2009**, 11, 4611.

[643] Di Meo, Florent; Trouillas, Pascal; Adamo, Carlo; Sancho-García, Juan C. Application of Recent Double-Hybrid Density Functionals to Low-Lying Singlet-Singlet Excitation Energies of Large Organic Compounds. *J. Chem. Phys.*, **2013**, 139, 164104. DOI: 10.1063/1.4825359.

[644] Hernández-Martínez, Laura; Brémond, Eric; Pérez-Jiménez, Angel J.; San-Fabián, Emilio; Adamo, Carlo; Sancho-García, Juan C. Nonempirical (double-hybrid) density functionals applied to atomic excitation energies: A systematic basis set investigation. *Int. J. Quantum Chem.*, **2020**, 120, e26193. DOI: 10.1002/qua.26193.

[645] DeBeer-George, S.; Petrenko, T.; Neese, F. *J. Phys. Chem. A*, **2008**, 112, 12936.

[646] Sørensen, Lasse Kragh; Guo, Meiyuan; Lindh, Roland; Lundberg, Marcus. Applications to metal K pre-edges of transition metal dimers illustrate the approximate origin independence for the intensities in the length representation. *Mol. Phys.*, **2017**, 115 (1-2), 174–189. DOI: 10.1080/00268976.2016.1225993.

[647] Bernadotte, Stephan; Atkins, Andrew J.; Jacob, Christoph R. Origin-Independent Calculation of Quadrupole Intensities in X-ray Spectroscopy. *J. Chem. Phys.*, **2012**, 137, 204106. DOI: 10.1063/1.4766359.

[648] Holmgaard List, N. Saue, T.; Norman, P. Rotationally averaged linear absorption spectra beyond the electric-dipole approximation. *Mol. Phys.*, **2017**, 115 (1-2), 63–74. DOI: 10.1080/00268976.2016.1187773.

[649] List, N. H.; Melin, T. R. L.; van Horn, M.; Saue, T. Beyond the electric-dipole approximation in simulations of x-ray absorption spectroscopy: Lessons from relativistic theory. *J. Chem. Phys.*, **2020**, 152 (18), 184110. DOI: 10.1063/5.0003103.

[650] Ray, K.; DeBeer-George, S.; Solomon, E. I.; Wieghardt, K.; Neese, F. *Chem. Eur. J.*, **2007**, 13, 2783.

[651] Steinmetzer, Johannes; Kupfer, Stephan; Gräfe, Stefanie. pysisyphus: Exploring potential energy surfaces in ground and excited states. *Int. J. Quantum Chem.*, **2021**, 121 (3), e26390. DOI: 10.1002/qua.26390.

[652] Campetella, Marco; Sanz García, Juan. Following the evolution of excited states along photochemical reaction pathways. *J. Comput. Chem.*, **2020**, 41 (12), 1156–1164. DOI: 10.1002/jcc.26162.

[653] Send, Robert; Furche, Filipp. First-Order Nonadiabatic Couplings from Time-Dependent Hybrid Density Functional Response Theory: Consistent Formalism, Implementation, and Performance. *J. Chem. Phys.*, **2010**, 132 (4), 044107. DOI: 10.1063/1.3292571.

[654] Fatehi, Shervin; Alguire, Ethan; Shao, Yihan; Subotnik, Joseph E. Analytic Derivative Couplings between Configuration-Interaction-Singles States with Built-in Electron-Translation Factors for Translational Invariance. *J. Chem. Phys.*, **2011**, 135 (23), 234105. DOI: 10.1063/1.3665031.

[655] Li, Zhendong; Liu, Wenjian. First-Order Nonadiabatic Coupling Matrix Elements between Excited States: A Lagrangian Formulation at the CIS, RPA, TD-HF, and TD-DFT Levels. *J. Chem. Phys.*, **2014**, 141 (1), 014110. DOI: 10.1063/1.4885817.

[656] Neese, F.; Solomon, E. I. *Inorg. Chem.*, **1998**, 37, 6568–6582.

[657] Maganas, D.; DeBeer, S.; Neese, F. Pair Natural Orbital Restricted Open-Shell Configuration Interaction (PNO-ROCIS) Approach for Calculating x-Ray Absorption Spectra of Large Chemical Systems. *J. Phys. Chem. A*, **2018**, 122 (5), 1215–1227. DOI: 10.1021/acs.jpca.7b10880.

[658] Plasser, F.; Wormit, M.; Dreuw, A. *J. Chem. Phys.*, **2014**, 141, 024106.

[659] Helmich-Paris, Benjamin. Simulating X-ray absorption spectra with complete active space self-consistent field linear response methods. *Int. J. Quantum Chem.*, **2021**, 121 (3), e26559. DOI: 10.1002/qua.26559.

[660] Olsen, Jeppe; Jensen, Hans Jørgen Aa.; Jørgensen, Poul. Solution of the large matrix equations which occur in response theory. *J. Comput. Phys.*, **1988**, 74 (2), 265 – 282. DOI: 10.1016/0021-9991(88)90081-2.

[661] Chaban, G.; Schmidt, M. W.; Gordon, M. S. Approximate second order method for orbital optimization of SCF and MCSCF wavefunctions. *Theor. Chem. Acc.*, **1997**, 97, 88–95. DOI: 10.1007/s002140050241.

[662] Dutta, Achintya Kumar; Neese, Frank; Izsák, Róbert. Speeding up Equation of Motion Coupled Cluster Theory with the Chain of Spheres Approximation. *J. Chem. Phys.*, **2016**, 144 (3), 034102.

[663] Nooijen, Marcel; Bartlett, Rodney J. A new method for excited states: Similarity transformed equation-of-motion coupled-cluster theory. *J. Chem. Phys.*, **1997**, 106 (15), 6441–6448. DOI: 10.1063/1.474000.

[664] Huntington, L .M. J.; Krupička, M.; Neese, F.; Izsák, R. Similarity transformed equation of motion coupled-cluster theory based on an unrestricted Hartree-Fock reference for applications to high-spin open-shell systems. *J. Chem. Phys.*, **2017**, 147, 174104.

[665] Casanova-Páez, M.; Neese, F. Assessment of the similarity-transformed equation of motion (STEOM) for open-shell organic and transition metal molecules. *J. Chem. Phys.*, **2024**, 161, 1444120–XXXX. DOI: 10.1063/5.0234225.

[666] Dutta, Achintya Kumar; Nooijen, Marcel; Neese, Frank; Izsák, Róbert. Towards a Pair Natural Orbital Coupled Cluster Method for Excited States. *J. Chem. Phys.*, **2017**, 146 (3), 034102. DOI: 10.1063/1.4974488.

[667] Ghosh, Soumen; Dutta, Achintya Kumar; de Souza, Bernardo; Berraud-Pache, Romain; Izsák, Róbert. A New Density for Transition Properties within the Similarity Transformed Equation of Motion Approach. *Mol. Phys.*, **2020**, 118 (19-20), e1818858. DOI: 10.1080/00268976.2020.1818858.

[668] Dutta, Achintya Kumar; Neese, Frank; Izsák, Róbert. Towards a Pair Natural Orbital Coupled Cluster Method for Excited States. *J. Chem. Phys.*, **2016**, 145 (3), 034102.

[669] Ghosh, Soumen; Dutta, Achintya Kumar; de Souza, Bernardo; Berraud-Pache, Romain; Róbert Izsák. A New Density for Transition Properties within the Similarity Transformed Equation of Motion Approach. *Mol. Phys.*, **2020**, 0 (0), e1818858. DOI: 10.1080/00268976.2020.1818858.

[670] DeBeer-George, S.; Petrenko, T.; Neese, F. *Inorg. Chim. Acta*, **2008**, 361, 965.

[671] Casanova-Páez, M.; Neese, F. Core-Excited States for Open-Shell Systems in Similarity-Transformed Equation-of-Motion Theory. *J. Chem. Theory Comput.*, **2025**, 21 (3), 1306–1321. DOI: 10.1021/acs.jctc.4c01181.

[672] Petrenko, Taras; Neese, Frank. Analysis and prediction of absorption band shapes, fluorescence band shapes, resonance Raman intensities, and excitation profiles using the time-dependent theory of electronic spectroscopy. *J. Chem. Phys.*, **2007**, 127 (16), 164319. DOI: 10.1063/1.2770706.

[673] Barone, Vincenzo; Biczysko, Malgorzata; Bloino, Julien. Fully Anharmonic IR and Raman Spectra of Medium-Size Molecular Systems: Accuracy and Interpretation. *Phys. Chem. Chem. Phys.*, **2014**, 16 (5), 1759–1787. DOI: 10.1039/C3CP53413H.

[674] Yagi, Kiyoshi; Hirao, Kimihiko; Taketsugu, Tetsuya; Schmidt, Michael W.; Gordon, Mark S. \emph Ab initio Vibrational State Calculations with a Quartic Force Field: Applications to H2CO, C2H4, CH3OH, CH3CCH, and C6H6. *J. Chem. Phys.*, **2004**, 121 (3), 1383–1389. DOI: 10.1063/1.1764501.

[675] Barnes, Loïc; Schindler, Baptiste; Compagnon, Isabelle; Allouche, Abdul-Rahman. Fast and Accurate Hybrid QM//MM Approach for Computing Anharmonic Corrections to Vibrational Frequencies. *J. Mol. Model.*, **2016**, 22 (11), 285. DOI: 10.1007/s00894-016-3135-5.

[676] Kesharwani, Manoj K.; Brauer, Brina; Martin, Jan M. L. Frequency and Zero-Point Vibrational Energy Scale Factors for Double-Hybrid Density Functionals (and Other Selected Methods): Can Anharmonic Force Fields Be Avoided? *J. Phys. Chem. A*, **2015**, 119 (9), 1701–1714. DOI: 10.1021/jp508422u.

[677] Bec, Krzysztof B.; Huck, Christian W. Breakthrough Potential in Near-Infrared Spectroscopy: Spectra Simulation. A Review of Recent Developments. *Front. Chem.*, **2019**, 7, 48. DOI: 10.3389/fchem.2019.00048.

[678] Reiter, Kevin; Kühn, Michael; Weigend, Florian. Vibrational circular dichroism spectra for large molecules and molecules with heavy elements. *J. Chem. Phys.*, **2017**, 146 (5), 054102. DOI: 10.1063/1.4974897.

[679] Neugebauer, J.; Reiher, M.; Kind, C.; Hess, B. A. *J. Comput. Chem.*, **2002**, 23, 895–910.

[680] Petrenko, Taras; Sturhahn, Wolfgang; Neese, Frank. *Hyperfine Interact.*, **2007**, 175, 165.

[681] Petrenko, Taras; DeBeer-George, Serena; Aliaga-Alcalde, Núria; Bill, Eckhard; Mienert, Bernd; Xiao, Yuming; Guo, YiSong; Sturhahn, Wolfgang; Cramer, Stephen P.; Wieghardt, Karl; Neese, Frank. Characterization of a Genuine Iron(V)-Nitrido Species by Nuclear Resonant Vibrational Spectroscopy Coupled to Density Functional Calculations. *J. Am. Chem. Soc.*, **2007**, 129, 11053–11060.

[682] McLean, A. D.; Yoshimine, M. Theory of Molecular Polarizabilities. *J. Chem. Phys.*, **1967**, 47 (6), 1927–1935. arXiv:https://pubs.aip.org/aip/jcp/article-pdf/47/6/1927/18852261/1927_1_online.pdf, DOI: 10.1063/1.1712220.

[683] Elking, Dennis M.; Perera, Lalith; Duke, Robert; Darden, Thomas; Pedersen, Lee G. A finite field method for calculating molecular polarizability tensors for arbitrary multipole rank. *J. Comput. Chem.*, **2011**, 32 (15), 3283–3295. DOI: 10.1002/jcc.21914.

[684] Chen, Houxian; Liu, Menglin; Yan, Tianying. Molecular multipoles and (hyper)polarizabilities from the Buckingham expansion: revisited. *Commun. Theor. Phys.*, **2020**, 72 (7), 075503. DOI: 10.1088/1572-9494/ab8a0d.

[685] London, F. Théorie quantique des courants interatomiques dans les combinaisons aromatiques. *Phys. Radium*, **1937**, 8 (10), 397–409. DOI: 10.1051/jphysrad:01937008010039700.

[686] Ditchfield, R. *J. Chem. Phys.*, **1972**, 56, 5688.

[687] Helgaker, T.; Jaszuński, M.; Ruud, K. *Chem. Rev.*, **1999**, 99, 293.

[688] Gauss, J. Molecular Properties. In Grotendorst, J., editor, *Modern Methods and Algorithms of Quantum Chemistry*, volume 3, 541–592. John von Neumann Institute for Computing, NIC Series, **2000**.

[689] Mason, J. Convention for the Reporting of Nuclear Magnetic Shielding (or Shifts) Tensors Suggested by Participants in the NATO ARW on NMR Shielding Constants at the University of Maryland, College Park, July 1992. *Solid State Nucl. Magn. Res.*, **1993**, 2, 285–288. DOI: 10.1016/0926-2040(93)90046-3.

[690] Auer, A. A.; Gauss, J., Stanton J. F. Quantitative prediction of gas-phase ^{13}C nuclear magnetic shielding constants. *J. Chem. Phys.*, **2003**, 118, 10407. DOI: 10.1063/1.1574314.

[691] Flaig, D.; Maurer, M.; Hanni, M.; Braunger, K.; Kick, L.; Thubauville, M.; Ochsenfeld, C. *J. Chem. Theory Comput.*, **2014**, 10, 572.

[692] Stoychev, Georgi L; Auer, Alexander A; Izsák, Róbert; Neese, Frank. Self-Consistent Field Calculation of Nuclear Magnetic Resonance Chemical Shielding Constants Using Gauge-Including Atomic Orbitals and Approximate Two-Electron Integrals. *J. Chem. Theory Comput.*, **2018**, 14 (2), 619–637. DOI: 10.1021/acs.jctc.7b01006.

[693] Maximoff, Sergey N.; Scuseria, Gustavo E. Nuclear Magnetic Resonance Shielding Tensors Calculated with Kinetic Energy Density-Dependent Exchange-Correlation Functionals. *Chem. Phys. Lett.*, **2004**, 390 (4-6), 408–412. DOI: 10.1016/j.cplett.2004.04.049.

[694] Schattenberg, Caspar Jonas; Kaupp, Martin. Effect of the Current Dependence of Tau-Dependent Exchange-Correlation Functionals on Nuclear Shielding Calculations. *J. Chem. Theory Comput.*, **2021**, 17 (3), 1469–1479. DOI: 10.1021/acs.jctc.0c01223.

[695] Dobson, John F. Alternative Expressions for the Fermi Hole Curvature. *J. Chem. Phys.*, **1993**, 98 (11), 8870–8872. DOI: 10.1063/1.464444.

[696] Bates, Jefferson E.; Furche, Filipp. Harnessing the Meta-Generalized Gradient Approximation for Time-Dependent Density Functional Theory. *J. Chem. Phys.*, **2012**, 137 (16), 164105. arXiv:23126693, DOI: 10.1063/1.4759080.

[697] Reimann, Sarah; Ekström, Ulf; Stopkowicz, Stella; Teale, Andrew M; Borgoo, Alex; Helgaker, Trygve. The Importance of Current Contributions to Shielding Constants in Density-Functional Theory. *Phys. Chem. Chem. Phys.*, **2015**, 17 (28), 18834–18842. arXiv:26123927, DOI: 10.1039/C5CP02682B.

[698] Van den Heuvel, Willem; Soncini, Alessandro. NMR Chemical Shift as Analytical Derivative of the Helmholtz Free Energy. *J. Chem. Phys.*, **2013**, 138, 054113. DOI: doi:10.1063/1.4789398.

[699] Soncini, Alessandro; Van den Heuvel, Willem. Communication: Paramagnetic NMR Chemical Shift in a Spin State Subject to Zero-Field Splitting. *J. Chem. Phys.*, **2013**, 138, 021103. DOI: doi:10.1063/1.4775809.

[700] Gauss, Jürgen; Ruud, Kenneth; Helgaker, Trygve. Perturbation-dependent atomic orbitals for the calculation of spin-rotation constants and rotational g tensors. *J. Chem. Phys.*, **1996**, 105 (7), 2804–2812. DOI: 10.1063/1.472143.

[701] Pell, Andrew J.; Pintacuda, Guido; Grey, Clare P. Paramagnetic NMR in Solution and the Solid State. *Prog. Nucl. Magn. Reson. Spectrosc.*, **2019**, 111, 1–271. DOI: 10.1016/j.pnmrs.2018.05.001.

[702] Lang, Lucas; Ravera, Enrico; Parigi, Giacomo; Luchinat, Claudio; Neese, Frank. Solution of a Puzzle: High-Level Quantum-Chemical Treatment of Pseudocontact Chemical Shifts Confirms Classic Semiempirical Theory. *J. Phys. Chem. Lett.*, **2020**, 11 (20), 8735–8744.

[703] Saitow, Masaaki; Neese, Frank. Accurate Spin-Densities Based on the Domain-Based Local Pair-Natural Orbital Coupled-Cluster Theory. *J. Chem. Phys.*, **2018**, 149, 034104. DOI: 10.1063/1.5027114.

[704] Pantazis, D. A.; Orio, M.; Petrenko, T.; Zein, S.; Bill, E.; Lubitz, W.; Messinger, J.; Neese, F. *Chem. Eur. J.*, **2009**, 15, 5108.

[705] Harriman, John E. *Theoretical Foundations of Electron Spin Resonance: Physical Chemistry: A Series of Monographs*. Academic Press, **1978**. ISBN 978-1483175855.

[706] Pederson, M. R.; Khanna, S. N. *Phys. Rev. B*, **1999**, 60, 9566.

[707] Neese, F. *J. Chem. Phys.*, **2007**, 127, 164112.

[708] Sinnecker, S.; Neese, F. Spin-Spin Contributions to the Zero-Field Splitting Tensor in Organic Triplets, Carbenes and Biradicals – A Density Functional and \it ab initio Study. *J. Phys. Chem. A*, **2006**, 110, 12267.

[709] Riplinger, Christoph; Kao, Joseph P. Y.; Rosen, Gerald M.; Kathirvelu, Velavan; Eaton, Gareth R.; Eaton, Sandra S.; Kutateladze, Andrei; Neese, Frank. *J. Am. Chem. Soc.*, **2009**, 131, 10092.

[710] Perdew, J. P.; Chevary, J. A.; Vosko, S. H.; Jackson, K. A.; Pederson, M. R.; Singh, D. J.; Fiolhais, C. *Phys. Rev. A*, **1992**, 46, 6671.

[711] Helgaker, Trygve U.; Almlöf, Jan; Jensen, Hans Jørgen Aa.; Jørgensen, Poul. Molecular Hessians for large-scale MCSCF wave functions. *J. Chem. Phys.*, **1986**, 84 (11), 6266–6279. DOI: 10.1063/1.450771.

[712] Vahtras, O.; Minaev, B.; Ågren, H. Ab initio calculations of electronic g-factors by means of multiconfiguration response theory. *Chem. Phys. Lett.*, **1997**, 281 (1), 186–192. DOI: 10.1016/S0009-2614(97)01169-X.

[713] Neese, F. *Inorg. Chim. Acta*, **2002**, 337C, 181–192.

[714] Römelt, M.; Ye, S.; Neese, F. *Inorg. Chem.*, **2009**, 48, 784.

[715] Neese, F. Efficient and Accurate Approximations to the Molecular Spin-Orbit Coupling Operator and their use in Molecular g-Tensor Calculations. *J. Chem. Phys.*, **2005**, 122, 034107.

[716] Ganyushin, D.; Neese, F. *J. Chem. Phys.*, **2013**, 138, 104113.

[717] Hess, Bernd A.; Marian, Christel M.; Wahlgren, Ulf; Gropen, Odd. *Chem. Phys. Lett.*, **1996**, 251, 365–371.

[718] Schimmelpfennig, B. AMFI - an Atomic Mean-Field Spin-Orbit Integral Program. **1996**.

[719] Berning, A.; Schweizer, M.; Werner, H.J.; Knowles, P. J.; Palmieri, P. Spin-orbit matrix elements for internally contracted multireference configuration interaction wavefunctions. *Mol. Phys.*, **2000**, 98, 1823–1833. DOI: 10.1080/00268970009483386.

[720] Koseki, S.; Schmidt, M. W.; Gordon, M. S. *J. Phys. Chem.*, **1992**, 96, 10768–10772.

[721] Koseki, S.; Gordon, M. S.; Schmidt, M. W.; Matsunaga, N. Main-group effective nuclear charges for spin-orbit calculations. *J. Phys. Chem.*, **1995**, 99 (31), 12764–12772. DOI: 10.1021/j100034a013.

[722] Koseki, S.; Schmidt, M. W.; Gordon, M. S. Reappraisal of the spin-forbidden unimolecular decay of the methoxy radical. *J. Phys. Chem. A*, **1998**, 102 (50), 10430–10435. DOI: 10.1021/jp982609q.

[723] Suturina, E. A.; Maganas, D.; Bill, E.; Atanasov, M.; Neese, F. *Inorg. Chem.*, **2015**, 54, 9948–9961.

[724] Ginsberg, A. P. *J. Am. Chem. Soc.*, **1980**, 102, 111.

[725] Noodleman, L. *J. Chem. Phys.*, **1981**, 74, 5737.

[726] Noodleman, L.; Davidson, E. R. *Chem. Phys.*, **1986**, 109, 131.

[727] Bencini, A.; Gatteschi, D. X.alpha.-SW calculations of the electronic structure and magnetic properties of weakly coupled transition-metal clusters. The [Cu2Cl6]2- dimers. *J. Am. Chem. Soc.*, **1980**, 108, 5763. DOI: 10.1021/ja00279a017.

[728] Yamaguchi, K.; Takahara, Y.; Fueno, T. In Smith, V. H., editor, *Applied Quantum Chemistry*, pages 155. Wiley, **1986**.

[729] Soda, T.; Kitagawa, Y.; Onishi, T.; Takano, Y.; Shigeta, Y.; Nagao, H.; Yoshioka, Y.; Yamaguchi, K. *Chem. Phys. Lett.*, **2000**, 319, 223.

[730] Yamaguchi, K.; Jensen, F.; Dorigo, A.; Houk, K. N. *Chem. Phys. Lett.*, **1988**, 149, 537.

[731] Saito, T.; Nishihara, S.; Kataoka, Y.; Nakanishi, Y.; Kitagawa, Y.; Kawakami, T.; Yamanaka, S.; Okumura, M.; Yamaguchi, K. *J. Phys. Chem. A*, **2010**, 114, 7967.

[732] Ruiz, E.; Cano, J.; Alvarez, S.; Alemany, P. *J. Comput. Chem.*, **1999**, 20, 1391.

[733] Saito, T.; Thiel, W. *J. Phys. Chem. A*, **2012**, 116, 10864.

[734] Coulaud, E.; Malrieu, J.-P.; Guihéry, N.; Ferré, N. Additive Decomposition of the Physical Components of the Magnetic Coupling from Broken Symmetry Density Functional Theory Calculations. *J. Chem. Theory Comput.*, **2013**, 9 (8), 3429–3436. DOI: 10.1021/ct400305h.

[735] Ferre, N.; Guihery, N.; Malrieu, J.-P. Spin Decontamination of Broken-Symmetry Density Functional Theory Calculations: Deeper Insight and New Formulations. *Phys. Chem. Chem. Phys.*, **2015**, 17 (22), 14375–14382. DOI: 10.1039/C4CP05531D.

[736] David, Grégoire; Trinquier, Georges; Malrieu, Jean-Paul. Consistent spin decontamination of broken-symmetry calculations of diradicals. *J. Chem. Phys.*, **2020**, 153 (19), 194107. DOI: 10.1063/5.0029201.

[737] Duplaix-Rata, Gwenhaël; Le Guennic, Boris; David, Grégoire. Revisiting magnetic exchange couplings in heterodinuclear complexes through the decomposition method in KS-DFT. *Phys. Chem. Chem. Phys.*, **2023**, 25 (20), 14170–14178. DOI: 10.1039/d3cp00697b.

[738] David, Grégoire; Ferré, Nicolas; Le Guennic, Boris. Consistent Evaluation of Magnetic Exchange Couplings in Multicenter Compounds in KS-DFT: The Recomposition Method. *J. Chem. Theory Comput.*, **2022**, 19 (1), 157–173. DOI: 10.1021/acs.jctc.2c01022.

[739] David, Grégoire; Duplaix-Rata, Gwenhaël; Le Guennic, Boris. What governs magnetic exchange couplings in radical-bridged dinuclear complexes? *Phys. Chem. Chem. Phys.*, **2024**, 26 (11), 8952–8964. DOI: 10.1039/d3cp06243k.

[740] Lunghi, Alessandro; Totti, Federico; Sanvito, Stefano; Sessoli, Roberta. Intra-molecular origin of the spin-phonon coupling in slow-relaxing molecular magnets. *Chem. Sci.*, **2017**, 8 (9), 6051–6059.

[741] Atanasov, M.; Aravena, D.; Suturina, E.; Bill, E.; Maganas, D.; Neese, F. First principles approach to the electronic structure, magnetic anisotropy and spin relaxation in mononuclear 3d-transition metal single molecule magnets. *Coord. Chem. Rev.*, **2015**, 289, 177–214.

[742] Reta, Daniel; Kragskow, Jon GC; Chilton, Nicholas F. Ab initio prediction of high-temperature magnetic relaxation rates in single-molecule magnets. *Journal of the American Chemical Society*, **2021**, 143 (15), 5943–5950.

[743] Kragskow, Jon GC; Mattioni, Andrea; Staab, Jakob K; Reta, Daniel; Skelton, Jonathan M; Chilton, Nicholas F. Spin–phonon coupling and magnetic relaxation in single-molecule magnets. *Chemical Society Reviews*, **2023**, 52 (14), 4567–4585.

[744] Mariano, Lorenzo A; Nguyen, Vu Ha Anh; Petersen, Jonatan B; Björnsson, Magnus; Bendix, Jesper; Eaton, Gareth R; Eaton, Sandra S; Lunghi, Alessandro. The role of electronic excited states in the spin-lattice relaxation of spin-1/2 molecules. *Science Advances*, **2025**, 11 (7), eadr0168.

[745] Timm, Carsten. Time-convolutionless master equation for quantum dots: Perturbative expansion to arbitrary order. *Physical Review B—Condensed Matter and Materials Physics*, **2011**, 83 (11), 115416.

[746] Lunghi, Alessandro; Sanvito, Stefano. The limit of spin lifetime in solid-state electronic spins. *The Journal of Physical Chemistry Letters*, **2020**, 11 (15), 6273–6278.

[747] Lunghi, Alessandro; Sanvito, Stefano. How do phonons relax molecular spins? *Science advances*, **2019**, 5 (9), eaax7163.

[748] Lunghi, Alessandro. Toward exact predictions of spin-phonon relaxation times: An ab initio implementation of open quantum systems theory. *Science Advances*, **2022**, 8 (31), eabn7880.

[749] Foglia, Nicolás O.; Maganas, Dimitrios; Neese, Frank. Going beyond the electric-dipole approximation in the calculation of absorption and (magnetic) circular dichroism spectra including scalar relativistic and spin–orbit coupling effects. *J. Chem. Phys.*, **2022**, 157 (8), 084120. arXiv:10.1063/5.0094709, DOI: 10.1063/5.0094709.

[750] Chibotaru, L. F.; Ungur, L. Ab Initio Calculation of Anisotropic Magnetic Properties of Complexes. I. Unique Definition of Pseudospin Hamiltonians and Their Derivation. *J. Chem. Phys.*, **2012**, 137 (6), 064112. DOI: 10.1063/1.4739763.

[751] Ungur, Liviu; Chibotaru, Liviu F. Ab Initio Crystal Field for Lanthanides. *Chem. Eur. J.*, **2017**, 23 (15), 3708–3718. DOI: 10.1002/chem.201605102.

[752] Iwahara, Naoya; Ungur, Liviu; Chibotaru, Liviu F. J-Pseudospin States and the Crystal Field of Cubic Systems. *Phys. Rev. B*, **2018**, 98 (5), 054436. DOI: 10.1103/PhysRevB.98.054436.

[753] Ungur, Liviu. *Ab Initio Methodology for the Investigation of Magnetism in Strongly Anisotropic Complexes*. PhD thesis, KU Leuven, 10 **2010**.

[754] Ungur, Liviu. Introduction to the Electronic Structure, Luminescence, and Magnetism of Lanthanides. In Martín-Ramos, Pablo; Manuela Ramos Silva, editors, *Lanthanide-Based Multifunctional Materials*, Advanced Nanomaterials, pages 1–58. Elsevier, **2018**. DOI: 10.1016/B978-0-12-813840-3.00001-6.

[755] Vieru, Veacheslav; Iwahara, Naoya; Ungur, Liviu; Chibotaru, Liviu F. Giant exchange interaction in mixed lanthanides. *Scientific reports*, **2016**, 6, 24046.

[756] Iwahara, Naoya; Chibotaru, Liviu F. Exchange interaction between J multiplets. *Physical Review B*, **2015**, 91, 174438. DOI: 10.1103/PhysRevB.91.174438.

[757] Lines, M. E. Orbital Angular Momentum in the Theory of Paramagnetic Clusters. *J. Chem. Phys.*, **1971**, 55 (6), 2977–2984. DOI: 10.1063/1.1676524.

[758] Schmerwitz, Yorick Leonard Adrian; Ivanov, Aleksei V.; Jónsson, Elvar Ö.; Jónsson, Hannes; Levi, Gianluca. Variational Density Functional Calculations of Excited States: Conical Intersection and Avoided Crossing in Ethylene Bond Twisting. *J. Phys. Chem. Lett.*, **2022**, 13, 3990–3999. DOI: 10.1021/acs.jpclett.2c00741.

[759] Selenius, Elli; Sigurdarson, Alec Elías; Schmerwitz, Yorick Leonard Adrian; Levi, Gianluca. Orbital-Optimized Versus Time-Dependent Density Functional Calculations of Intramolecular Charge Transfer Excited States. *J. Chem. Theory Comput.*, **2024**, 20, 3809–3822. DOI: 10.1021/acs.jctc.3c01319.

[760] Gilbert, Andrew T. B.; Besley, Nicholas A.; Gill, Peter M. W. Self-Consistent Field Calculations of Excited States Using the Maximum Overlap Method (MOM). *J. Phys. Chem. A*, **2008**, 112 (50), 13164–13171. Publisher: American Chemical Society. DOI: 10.1021/jp801738f.

[761] Corzo, Hector H.; Abou Taka, Ali; Pribram-Jones, Aurora; Hratchian, Hrant P. Using projection operators with maximum overlap methods to simplify challenging self-consistent field optimization. *J. Comput. Chem.*, **2022**, 43 (6), 382–390. DOI: 10.1002/jcc.26797.

[762] Hait, Diptarka; Head-Gordon, Martin. Orbital Optimized Density Functional Theory for Electronic Excited States. *J. Phys. Chem. Lett.*, **2021**, 12, 4517–4529. DOI: 10.1021/acs.jpclett.1c00744.

[763] Hardikar, Tarini S.; Neuscamman, Eric. A self-consistent field formulation of excited state mean field theory. *J. Chem. Phys.*, **2020**, 153, 164108. DOI: 10.1063/5.0019557.

[764] Ziegler, Tom; Rank, Arvi; Baerends, Evert J. On the Calculation of Multiplet Energies by the Hartree-Fock-Slater Method. *Theoret. Chim Acta (Berl.)*, **1977**, 43, 261–271. DOI: 10.1007/BF00551551.

[765] Schmerwitz, Yorick Leonard Adrian; Levi, Gianluca; Jónsson, Hannes. Calculations of Excited Electronic States by Converging on Saddle Points Using Generalized Mode Following. *J. Chem. Theory Comput.*, **2023**, 19, 3634–3651. DOI: 10.1021/acs.jctc.3c00178.

[766] Schmerwitz, Yorick Leonard Adrian; Ollé, Núria Urgell; Levi, Gianluca; Jónsson, Hannes. Saddle Point Search Algorithms for Variational Density Functional Calculations of Excited Electronic States with Self-Interaction Correction. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, 1–11. Association for Computing Machinery, 6 **2024**. DOI: 10.1145/3659914.3659933.

[767] Schmerwitz, Yorick Leonard Adrian; Selenius, Elli; Levi, Gianluca. Freeze-and-release direct optimization method for variational calculations of excited electronic states. *arXiv:2501.18568*, **2025**. DOI: 10.48550/arXiv.2501.18568.

[768] Levi, Gianluca; Ivanov, Aleksei V; Jónsson, Hannes. Variational density functional calculations of excited states via direct optimization. *J. Chem. Theory Comput.*, **2020**, 16 (11), 6968–6982. DOI: 10.1021/acs.jctc.0c00597.

[769] Barca, Giuseppe M. J.; Gilbert, Andrew T. B.; Gill, Peter M. W. Simple Models for Difficult Electronic Excitations. *J. Chem. Theory Comput.*, **2018**, 14 (3), 1501–1509. Publisher: American Chemical Society. DOI: 10.1021/acs.jctc.7b00994.

[770] Carter-Fenk, Kevin; Herbert, John M. State-Targeted Energy Projection: A Simple and Robust Approach to Orbital Relaxation of Non-Aufbau Self-Consistent Field Solutions. *J. Chem. Theory Comput.*, **2020**, 16 (8), 5067–5082. Publisher: American Chemical Society. DOI: 10.1021/acs.jctc.0c00502.

[771] Kubas, Adam; Hoffmann, Felix; Heck, Alexander; Oberhofer, Harald; Elstner, Marcus; Blumberger, Jochen. Electronic couplings for molecular charge transfer: Benchmarking CDFT, FODFT, and FODFTB against high-level ab initio calculations. *J. Chem. Phys.*, **2014**, 140 (10), 104105. DOI: 10.1063/1.4867077.

[772] Mitoraj, Mariusz P.; Michalak, Artur; Ziegler, Tom. A Combined Charge and Energy Decomposition Scheme for Bond Analysis. *J. Chem. Theory Comput.*, **2009**, 5 (4), 962–975.

[773] Schneider, W.; Bistoni, G.; Sparta., M.; Riplinger, C.; Saitow, M.; Auer, A.; Neese, F. Decomposition of Intermolecular Interaction Energies within the Local Pair Natural Orbital Coupled Cluster Framework. *J. Chem. Theory Comput.*, **2016**, 12 (10), 4778–4792. DOI: 10.1021/acs.jctc.6b00523.

[774] Altun, Ahmet; Saitow, Masaaki; Neese, Frank; Bistoni, Giovanni. Local Energy Decomposition of Open-Shell Molecular Systems in the Domain-Based Local Pair Natural Orbital Coupled Cluster Framework. *J. Chem. Theory Comput.*, **2019**, 15 (3), 1616–1632. DOI: 10.1021/acs.jctc.8b01145.

[775] Bistoni, Giovanni. Finding Chemical Concepts in the Hilbert Space: Coupled Cluster Analyses of Noncovalent Interactions. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, **2020**, 10 (3), e1442. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1442, DOI: 10.1002/wcms.1442.

[776] Bistoni, Giovanni; Auer, Alexander A.; Neese, Frank. Understanding the Role of Dispersion in Frustrated Lewis Pairs and Classical Lewis Adducts: A Domain-Based Local Pair Natural Orbital Coupled Cluster Study. *Chem. Eur. J.*, **2017**, 23 (4), 865–873.

[777] Lu, Qing; Neese, Frank; Bistoni, Giovanni. Formation of Agostic Structures Driven by London Dispersion. *Angew. Chem. Int. Ed.*, **2018**, 57 (17), 4760–4764.

[778] Lu, Qing; Neese, Frank; Bistoni, Giovanni. London Dispersion Effects in the Coordination and Activation of Alkanes in σ-Complexes: A Local Energy Decomposition Study. *Phys. Chem. Chem. Phys.*, **2019**, 21 (22), 11569–11577. DOI: 10.1039/C9CP01309A.

[779] Ghafarian Shirazi, Reza; Neese, Frank; Pantazis, Dimitrios A; Bistoni, Giovanni. Physical Nature of Differential Spin-State Stabilization of Carbenes by Hydrogen and Halogen Bonding: A Domain-Based Pair Natural Orbital Coupled Cluster Study. *J. Phys. Chem. A*, **2019**, 123 (24), 5081–5090.

[780] Yepes, Diana; Neese, Frank; List, Benjamin; Bistoni, Giovanni. Unveiling the Delicate Balance of Steric and Dispersion Interactions in Organocatalysis Using High-Level Computational Methods. *J. Am. Chem. Soc.*, **2020**, 142 (7), 3613–3625. arXiv:10.1021/jacs.9b13725, DOI: 10.1021/jacs.9b13725.

[781] Beck, M. E.; Riplinger, C.; Neese, F.; Bistoni, G. Unraveling Individual Host-Guest Interactions in Molecular Recognition from First Principles Quantum Mechanics: Insights into the Nature of Nicotinic Acetylcholine Receptor Agonist Binding. *J. Comput. Chem.*, **2021**, 42 (5), 293–302. DOI: 10.1002/jcc.26454.

[782] Altun, Ahmet; Izsák, Róbert; Bistoni, Giovanni. Local Energy Decomposition of Coupled-Cluster Interaction Energies: Interpretation, Benchmarks, and Comparison with Symmetry-Adapted Perturbation Theory. *Int. J. Quantum Chem.*, **2021**, 121 (3), e26339. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.26339, DOI: 10.1002/qua.26339.

[783] Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. Local Energy Decomposition Analysis of Hydrogen-Bonded Dimers within a Domain-Based Pair Natural Orbital Coupled Cluster Study. *Beilstein J. Org. Chem.*, **2018**, 14, 919. DOI: 10.1021/acs.jctc.8b01145.

[784] Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. Effect of Electron Correlation on Intermolecular Interactions: A Pair Natural Orbitals Coupled Cluster Based Local Energy Decomposition Study. *J. Chem. Theory Comput.*, **2019**, 15 (1), 215–228. DOI: 10.1021/acs.jctc.8b00915.

[785] Wuttke, Axel; Mata, Ricardo A. Visualizing Dispersion Interactions through the Use of Local Orbital Spaces. *J. Comput. Chem.*, **2017**, 38 (1), 15–23.

[786] Regni, Gianluca; Baldinelli, Lorenzo; Bistoni, Giovanni. A Quantum Chemical Method for Dissecting London Dispersion Energy into Atomic Building Blocks (In press). **2025**. DOI: 10.1021/acscentsci.5c00356.

[787] Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. HFLD: A Nonempirical London Dispersion-Corrected Hartree–Fock Method for the Quantification and Analysis of Noncovalent Interaction Energies of Large Molecular Systems. *J. Chem. Theory Comput.*, **2019**, 15 (11), 5894–5907. DOI: 10.1021/acs.jctc.9b00425.

[788] Baldinelli, Lorenzo; De Angelis, Filippo; Bistoni, Giovanni. Unraveling Atomic Contributions to the London Dispersion Energy: Insights into Molecular Recognition and Reactivity. *J. Chem. Theory Comput.*, **2024**, 20 (5), 1923–1931. DOI: 10.1021/acs.jctc.3c00977.

[789] Ghosh, Soumen; Neese, Frank; Izsák, Róbert; Bistoni, Giovanni. Fragment-Based Local Coupled Cluster Embedding Approach for the Quantification and Analysis of Noncovalent Interactions: Exploring the Many-Body Expansion of the Local Coupled Cluster Energy. *J. Chem. Theory Comput.*, **2021**, 17 (6), 3348–3359. DOI: 10.1021/acs.jctc.1c00005.

[790] Altun, Ahmet; Garcia-Ratés, Miquel; Neese, Frank; Bistoni, Giovanni. Unveiling the complex pattern of intermolecular interactions responsible for the stability of the DNA duplex. *Chem. Sci.*, **2021**, 12 (38), 12785–12793. DOI: 10.1039/D1SC03868K.

[791] Schümann, Jan M.; Ochmann, Lukas; Becker, Jonathan; Altun, Ahmet; Harden, Ingolf; Bistoni, Giovanni; Schreiner, Peter R. Exploring the Limits of Intramolecular London Dispersion Stabilization with Bulky Dispersion Energy Donors in Alkane Solution. *J. Am. Chem. Soc.*, **2023**, 145 (4), 2093–2097. DOI: 10.1021/jacs.2c13301.

[792] Altun, Ahmet; Schiavo, Eduardo; Mehring, Michael; Schulz, Stephan; Bistoni, Giovanni; Auer, Alexander A. Rationalizing polymorphism with local correlation-based methods: a case study of pnictogen molecular crystals. *Phys. Chem. Chem. Phys.*, **2024**, 26 (45), 28733–28745. DOI: 10.1039/D4CP03697B.

[793] Bistoni, Giovanni; Altun, Ahmet; Wang, Zikuan; Neese, Frank. Local Energy Decomposition Analysis of London Dispersion Effects: From Simple Model Dimers to Complex Biomolecular Assemblies. *Accounts Chem. Res.*, **2024**, 57 (9), 1411–1420. DOI: 10.1021/acs.accounts.4c00085.

[794] Altun, Ahmet; Leach, Isaac F.; Neese, Frank; Bistoni, Giovanni. A Generally Applicable Method for Disentangling the Effect of Individual Noncovalent Interactions on the Binding Energy. *Angew. Chem. Int. Ed.*, **2025**, 64 (12), e202421922. DOI: 10.1002/anie.202421922.

[795] Altun, Ahmet; Neese, Frank; Bistoni, Giovanni. HFLD: A Nonempirical London Dispersion-Corrected Hartree–Fock Method for the Quantification and Analysis of Noncovalent Interaction Energies of Large Molecular Systems. *J. Chem. Theory Comput.*, **2019**, 15 (11), 5894–5907. arXiv:10.1021/acs.jctc.9b00425, DOI: 10.1021/acs.jctc.9b00425.

[796] Altun, A.; Neese, F.; Bistoni, G. Open-Shell Variant of the London Dispersion-Corrected Hartree-Fock Method HFLD for the Quantification and Analysis of Noncovalent Interaction Energies. *J. Chem. Theory Comput.*, **2022**, 18 (4), 2292–2307. DOI: 10.1021/acs.jctc.1c01295.

[797] Greengard, L; Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.*, **1987**, 73 (2), 325–348. DOI: 10.1016/0021-9991(87)90140-9.

[798] Mayhall, Nicholas J.; Raghavachari, Krishnan; Hratchian, Hrant P. ONIOM-Based QM:QM Electronic Embedding Method Using Löwdin Atomic Charges: Energies and Analytic Gradients. *J. Chem. Phys.*, **2010**, 132 (11), 114107. DOI: 10.1063/1.3315417.

[799] Vreven, T.; Mennucci, B.; da Silva, C. O.; Morokuma, K.; Tomasi, J. *J. Chem. Phys.*, **2001**, 115, 62.

[800] Bjornsson, Ragnar; Bühl, Michael. Modeling Molecular Crystals by QM/MM: Self-Consistent Electrostatic Embedding for Geometry Optimizations and Molecular Property Calculations in the Solid. *J. Chem. Theory Comput.*, **2012**, 8 (2), 498–508. DOI: 10.1021/ct200824r.

[801] Anisimov, Victor; Stewart, James JP. *Introduction to the Fast Multipole Method: Topics in Computational Biophysics, Theory, and Implementation*. CRC Press, **2019**.

[802] Helgaker, Trygve; Jørgensen, Poul; Olsen, Jeppe. *Molecular electronic-structure theory*, chapter 9. John Wiley & Sons, **2013**.

[803] Pérez-Jordá, José M; Yang, Weitao. A concise redefinition of the solid spherical harmonics and its use in fast multipole methods. *J. Chem. Phys.*, **1996**, 104 (20), 8003–8006.

[804] Laio, Alessandro; Parrinello, Michele. Escaping Free-Energy Minima. *Proc. Natl. Acad. Sci. U.S.A.*, **2002**, 99 (20), 12562–12566. DOI: 10.1073/pnas.202427399.

[805] Iannuzzi, Marcella; Laio, Alessandro; Parrinello, Michele. Efficient Exploration of Reactive Potential Energy Surfaces Using Car–Parrinello Molecular Dynamics. *Phys. Rev. Lett.*, **2003**, 90 (23), 238302. DOI: 10.1103/PhysRevLett.90.238302.

[806] Tummanapelli, Anil Kumar; Vasudevan, Sukumaran. Dissociation Constants of Weak Acids from Ab Initio Molecular Dynamics Using Metadynamics: Influence of the Inductive Effect and Hydrogen Bonding on pKa Values. *J. Phys. Chem. B*, **2014**, 118 (47), 13651–13657. DOI: 10.1021/jp5088898.

[807] Tummanapelli, Anil Kumar; Vasudevan, Sukumaran. Estimating Successive pKa Values of Polyprotic Acids from Ab Initio Molecular Dynamics Using Metadynamics: The Dissociation of Phthalic Acid and Its Isomers. *Phys. Chem. Chem. Phys.*, **2015**, 17 (9), 6383–6388. DOI: 10.1039/C4CP06000H.

[808] Barducci, Alessandro; Bussi, Giovanni; Parrinello, Michele. Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method. *Phys. Rev. Lett.*, **2008**, 100 (2), 020603. DOI: 10.1103/PhysRevLett.100.020603.

[809] Martyna, Glenn J.; Klein, Michael L.; Tuckerman, Mark. Nosé–Hoover Chains: The Canonical Ensemble via Continuous Dynamics. *J. Chem. Phys.*, **1992**, 97 (4), 2635–2643. DOI: 10.1063/1.463940.

[810] Martyna, Glenn J.; Tuckerman, Mark E.; Tobias, Douglas J.; Klein, Michael L. Explicit Reversible Integrators for Extended Systems Dynamics. *Mol. Phys.*, **1996**, 87 (5), 1117–1157. DOI: 10.1080/00268979600100761.

[811] Bussi, Giovanni; Donadio, Davide; Parrinello, Michele. Canonical Sampling through Velocity Rescaling. *J. Chem. Phys.*, **2007**, 126 (1), 014101. DOI: 10.1063/1.2408420.

[812] Kumar, Shankar; Rosenberg, John M.; Bouzida, Djamal; Swendsen, Robert H.; Kollman, Peter A. THE Weighted Histogram Analysis Method for Free-Energy Calculations on Biomolecules. I. The Method. *J. Comput. Chem.*, **1992**, 13 (8), 1011–1021. DOI: 10.1002/jcc.540130812.

[813] Kästner, Johannes; Senn, Hans Martin; Thiel, Stephan; Otte, Nikolaj; Thiel, Walter. QM/MM Free-Energy Perturbation Compared to Thermodynamic Integration and Umbrella Sampling: Application to an Enzymatic Reaction. *J. Chem. Theory Comput.*, **2006**, 2 (2), 452–461. DOI: 10.1021/ct050252w.

[814] Grimme, Stefan. Exploration of Chemical Compound, Conformer, and Reaction Space with Meta-Dynamics Simulations Based on Tight-Binding Quantum Chemical Calculations. *J. Chem. Theory Comput.*, **2019**, 15 (5), 2847–2862. DOI: 10.1021/acs.jctc.9b00143.

[815] Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. Molecular Dynamics with Coupling to an External Bath. *J. Chem. Phys.*, **1984**, 81 (8), 3684–3690. DOI: 10.1063/1.448118.

[816] Box, G. E. P.; Muller, Mervin E. A Note on the Generation of Random Normal Deviates. *Ann. Math. Statist.*, **1958**, 29 (2), 610–611. DOI: 10.1214/aoms/1177706645.

[817] Andersen, Hans C. Rattle: A "Velocity" Version of the Shake Algorithm for Molecular Dynamics Calculations. *J. Comput. Phys.*, **1983**, 52 (1), 24–34. DOI: 10.1016/0021-9991(83)90014-1.

[818] Kutteh, Ramzi. RATTLE Recipe for General Holonomic Constraints: Angle and Torsion Constraints. *CCP5 Newsletter*, **1998**, 46, 8–15. URL: https://www.ccp5.ac.uk/wp-content/uploads/2023/03/CCP5_Newsletter_1998_10_46.pdf.

[819] Wilson, E. B.; Decius, J. C.; Cross, P. C. *Molecular Vibrations – the Theory of Infrared and Raman Vibrational Spectra*. Dover Publications, **1955**.

[820] Knizia, Gerald. Intrinsic Atomic Orbitals: An Unbiased Bridge between Quantum Theory and Chemical Concepts. *J. Chem. Theory Comput.*, **2013**, 9 (11), 4834–4843. DOI: 10.1021/ct400687b.

[821] Derricotte, Wallace D; Evangelista, Francesco A. Localized Intrinsic Valence Virtual Orbitals as a Tool for the Automatic Classification of Core Excited States. *J. Chem. Theory Comput.*, **2017**, 13 (12), 5984–5999. DOI: 10.1021/acs.jctc.7b00493.

[822] Kubas, A.; Berger, D.; Oberhofer, H.; Maganas, D.; Reuter, K.; Neese, F. Surface Adsorption Energetics Studied with "Gold Standard" Wave Function-Based Ab Initio Methods: Small-Molecule Binding to TiO2(110). *J. Phys. Chem. Lett.*, **2016**, 7, 4207–4212.

INDEX

A

Absorption Spectrum via ESD, 764
ADC2, 753
additive, 1189
ADEX, 1150
ADFT, 262
Ahlrichs Basis Sets, 63
AIMD, 1213
AMFI, 1054
Analytic DFT, 262
Angular Momentum Limits and Nomenclature, 160
ANO Basis Sets, 71
Architecture of ORCA, 1505
ASA, 978
Atomic Decomposition Methods, 1150
    ADEX, 1152
    ADLD, 1150
    ADLD(D), 1161
    ADLD(D4), 1161
Atomic Dipole Moment and Polarizabilities, 1014
AutoAux, 81
AUTOCI, 342
AUTOCI Keywords, 355
Aux Basis Sets, 79
AuxC, 79
AuxJ, 79
AuxJK, 79

B

B3LYP-3c, 261
B97-3c, 260
Basis Set Extrapolation, 312
Basis Set Keywords, 92
Basis Sets, 60
Basis Sets Linear Dependencies, 90
BLOCK Interface, 464
Broken-Symmetry Wavefunctions and Exchange Couplings, 1058
BubblePole Approximation, 158
BUPO, 158

C

C-PCM, 127
C-PCM Computing Terms Options, 137
CABS, 79
Cartesian Index Conventions for EPR and NMR Tensors, 1026
CAS-DFT, 494
CASPT2, 490
CASSCF, 358
CASSCF Linear Response, 1046
CASSCF Theory, 359
CC2, 343
CC3, 343
CCSDT, 343
CCSDT-n, 343
CHELP Charges, 717
CHELPG, 716
Circular Dichroism, 1078
Circularly Polarized Spectroscopies via ESD, 786
CIS, 809
CNDO, 248
Collinear Spin-Flip TDA, 815
Combination Bands, 987
Common Errors and Problems, 10
Compound, 1243
Configuration Interaction, 292
Configuration Interaction Hierarchy, 292
Conical Intersections, 667
Core Excitations (*CC*), 918
COSMO, 138
Counterpoise Corrections, 107
Counterpoise Corrections Keywords, 115
Coupled Cluster, 292
Coupled Cluster and CI Theories (*MDCI*), 292
Coupled Cluster S-Diagnostic, 309
Coupled-Cluster, 357
Coupled-Cluster Density, 302
CP-SCF, 203
Crystal-QMMM, 1195

D

DCD-CAS, 584
def2 Basis Sets, 64
DeltaSCF, 1111
Density Functional Theory, 211
    Custom Functionals, 214

DFT, 211
Double-Hybrid Functionals, 223
GGA, 213
Global Double-Hybrid Functionals, 224
Global Hybrid Functionals, 217
Hybrid Functionals, 216
LibXC, 230
Local Functionals, 213
Range-Separated Double-Hybrid Functionals, 227
Range-Separated Hybrid Functionals, 220
DFT-3c, 258
DFT-D3, 235
DFT-D4, 235
DFT-NL, 243
Diagonal Born-Oppenheimer Correction, 664
DIIS, 53
Dispersion Corrections, 234
Distance Dependent Pre-Screening, 154
DKS Basis Sets, 74
DLPNO Keywords, 330
DLPNO-EOM-CC, 762
DLPNO-NEVPT2-F12, 478
DLPNO-STEOM-CCSD, 907
DMRG, 464
DOCKER, 696
Double-Hybrid DFT and MP2, 284
Douglas-Kroll-Hess (*DKH*), 120
DRACO, 141
Dunning Basis Sets, 68
Dynamic Correlation, 304

E

EDA, 1124
EDA Keywords, 1134
Effective Core Potentials (*ECPs*), 81
Electric Moments, 1006
Electrical Properties, 1006
Electrical Properties Keywords, 1015
EOM-CCSD, 886
EOM-CCSD Keywords, 893
EOM-CCSD Theory, 886
EPR, 1030
EPR Tensors Index Conventions, 1026
ESD and STEOM-CCSD, 786
Essential Calculation Elements, 22
Excited State Dynamics (*ESD*), 764
Excited States Calculations, 742
Excited States via DFT, 815
Excited States via EOM-CCSD, 886
Excited States via HF, 814
Excited States via IH-FSMR, 903
Excited States via PNO-CC, 906
Excited States via Semiempirical Methods, 811
Excited States via STEOM, 895
Excited States via TDDFT, 809

F

F12 Basis Sets, 71
F12-CCSD(T), 310
F12-MP2, 310
Fast Multipole Method (*FMM*), 1205
FCI, 588
Fermi Smearing, 57
FIC-MRCC, 352
FIC-MRCI, 350
FIC-NEVPT, 353
Fluorescence Rates via ESD, 771
Fluorescence Spectra via ESD, 771
Foreword by F. Neese, i
Fractional Occupation, 58
Fractional Occupation Number Weighted Density (*FOD*), 736
Frontier MO Population Analysis, 712
Frozen Core, 201
Frozen Core Options MDCI, 301
FT-DFT, 57
Full-CI, 588

G

G-Tensor, 1031
G-Tensor Keywords, 1033
g-Tensor with MRCI, 534
Geometry Optimization, 601
GFN-xTB (*interfaced*), 251
GFN-xTB (*native*), 255
Ghost Atoms - Basis Sets, 85
GOAT, 673
Graphical User Interfaces (*GUI*), 2
GVPT2, 1000

H

Hartree-Fock Theory (*HF*), 210
HF-3c, 258
HFLD, 1158
HFLD Keywords, 1161
High Accuracy Thermochemistry: the HEAT protocol, 665
Hirshfeld Population Analysis, 713
Hyperfine Coupling, 1034
Hyperfine Coupling Keywords, 1038

I

ICE-CI, 446
ICE-NEVPT2, 481
IH-FSMR, 903
IH-FSMR-CCSD, 758
Implicit Solvation, 126
INDO, 248
Infrared Spectra, 986
Initial Guess, 197
Installation, 1

Integral Compression and Storage, 153
Integral Handling, 151
Interfaces, 1201
Internal Conversion Rates via ESD, 782
Intersystem Crossing Rates via ESD, 777
IR, 986
IRC, 631
Isomer Shift, 1052
Isotope Shifts in Vibrational Spectra, 999

J

Jensen Basis Sets, 65

K

KDIIS, 54

L

L-Edges, 861
LANL Basis Sets, 68
Literature Computational Chemistry, 6
Local Correlation, 317
Local Energy Decomposition (*LED*), 1134
Local Spin Analysis, 718
Löwdin Population Analysis, 711

M

Magnetic Circular Dichroism, 1078
Magnetic Circular Dichroism via ESD, 795
Magnetic Exchange Coupling Decomposition, 1060
Magnetic Exchange Properties, 1099
Magnetic Properties via MP2, 1045
Magnetic Properties via QDPT, 1061
Magnetic Properties via ROCIS, 875
Magnetic Properties with MRCI, 538
Magnetic Relaxation, 1071
Magrelax Keywords, 1077
Mayer Population Analysis, 712
MBIS, 714
MBIS Population Analysis, 714
MC-RPA, 750
MCD Keywords, 1081
MCD with MRCI, 540
MCPDFT, 494
MCRPA, 878
MCRPA Keywords, 886
MDCI, 292
MDCI Keywords, 330, 339
MECP, 670
meta-GGA, 213
MM, 592
MNDO, 248
Modelchemistries, 206
Molecular Dynamics, 1211
Molecular Dynamics (*MD*), 1213
Molecular Mechanics, 592
MP2 Derivatives, 272
MP2 Local, 275
MP2 Magnetic Properties, 1045
MP3, 343
MP4, 343
MP5, 343
MR-CI Tutorial, 503
MR-EOM-CC, 557
MRCI, 498
MRCI Keywords, 551
MRPT, 498
Mulliken Population Analysis, 709
Multiscale Simulations, 1176
Mössbauer Spectroscopy, 1051
Mössbauer Spectroscopy Keywords, 1054
Møller-Plesset Perturbation Theory (*MP2*), 266

N

Natural Bond Orbital Analysis, 725
Natural Chemical Shielding Analysis, 736
Natural Population Analysis, 713
Natural Transition Orbitals (*MCRPA*), 881
NBO, 725
NBO Analysis, 713
NCS, 736
NDDO, 248
NDO, 248
Near IR (*NIR*), 989
NEARIR, 987
NEB, 634
NEVPT2, 472
NEVPT2 Keywords, 488
NEVPT2-F12, 478
NICS, 1024
NL, 243
NMR, 1016
NMR Keywords EPRNMR, 1027
NMR NCS, 736
NMR Shielding Tensor Orbital Decomposition, 1025
NMR Spin-Spin Coupling Constants, 1019
NMR Tensors Index Conventions, 1026
NOCV, 1124
NOCV Keywords, 1134
Non-Adiabatic Coupling Matrix Elements, 853
NRVS Spectra, 994
NTOs and NDOs via ROCIS, 876
NTOs in TD-DFT, 831
Nuclear Magnetic Resonance, 1016
Nucleus-independent chemical shielding, 1024

O

One Photon Spectroscopy, 909

ONIOM, 1177
OO-MP2, 288
openCOSMO-RS, 142
OPI, 1243
ORCA citations, v
ORCA Name, 1
ORCA Python Interface, 1243
orca_2json, 1415
orca_mapspc, 1349
Overtones, 987

P

Paramagnetic NMR Shielding Tensors, 1029
PBEh-3c, 260
Phosphorescence Rates via ESD, 774
Phosphorescence Spectrum via ESD, 774
Picture Change Effects with MRCI, 545
PictureChange, 124
PNMR, 1029
PNO-CC, 760
PNO-EOM-CC, 906
Polarizabilities, 1006
POLY ANISO Interface, 1099
Population Analysis, 709
Population Analysis Keywords, 723
Population Analysis of Excited States (*TDDFT*), 820
Population Analysis of UNOs, 723
Post-HF Magnetic Properties, 1045
Potential Energy Surface Scans via TD-DFT, 845
Property File, 1436
Pseudospin Hamiltonians, 1082
PTE, 146
PTES, 146

Q

QDPT, 1061
QM/QM2, 1189
QM/QM2/MM, 1189
QMMM, 1177, 1189, 1201
QRO, 210
Quadrupole Splitting, 1053
Quasi-Degenerate SC-NEVPT2, 485
Quickstart Guide, 1

R

Raman Spectra, 992
Random Phase Approximation, 265
Relaxation, 1071
Resolution-of-the-Identity (*RI*), 94
Resolution-of-the-Identity Keywords, 102
Resonance Raman Spectra, 994
Resonant Raman Spectrum via ESD, 783
RESP, 717
RESP Charges, 717
RI-J, 95
RI-JK, 101
RI-MP2, 268
RIJCOSX, 97
RIXS via ROCIS, 868
ROCIS, 747, 854
ROCIS Keywords, 877
ROCIS Magnetic Properties, 875
RPA, 265, 809
RRHO, 659
Running ORCA, 3
r^2SCAN-3c, 260

S

Sapporo Basis Sets, 67
SARC Basis Sets, 76
SCF Convergence, 48
SCF Convergence Issues, 58
SCF Damping, 52
SCF Keywords, 56
SCF Level Shift, 53
Self-Consistent-Field (*SCF*), 47
Semiempirical Methods, 247
SF-TD-DFT, 815
SF-TDA, 809
Simplified TDA and TD-DFT, 823
SINGLE ANISO Interface, 1082
SMD, 138
SOLVATOR, 686
SOMF, 1054
SOSCF, 54
Spectroscopy & Properties, 708
Spin-Orbit Coupling, 1054
Spin-Orbit Coupling Correction to the Total Energy, 665
Spin-Rotation Constants, 1030
Spin-Spin Coupling Constants, 1019
srDFT, 495
Stability Analysis, 167
starting ORCA, 3
State-averaged SC-NEVPT2, 484
Static Correlation, 304
STEOM, 895
STEOM-CCSD, 755
STEOM-CCSD and ESD, 786
Structure and Reactivity, 600
subtractive, 1189
Symmetry, 191

T

TCutPairs, 330
TCutPNO, 330
TD-DFT, 742, 809
TD-DFT Keywords, 854
TD-DFT NTOs, 831
Thermochemistry, 659
TRAH, 55

Transition Metal L-Edges with ROCIS, 861
Transition States, 623

U

UCO, 210
UHF Natural Orbitals, 210
UNO, 210
Utilities, 1348
Utilities and Visualization, 1338

V

Vibrational Circular Dichroism, 991
Vibrational Corrections via VPT2, 1002
Vibrational Frequencies, 653
Vibrational Spectroscopy, 986
VPT2, 1000

W

Why is ORCA called ORCA ?, 1
Workflows and Automatization, 1241

X

X2C, 121
X-Ray Spectroscopy with MRCI, 546
XAS via MCRPA, 883
XES, 1162

Z

Zero-Field Splitting, 1039
Zero-Field Splitting Keywords, 1045
Zero-Field Splitting with MRCI, 530
ZORA, 118
ZORA Basis Sets, 74

⊡

ΔSCF, 1111

⊡

ωB97X-3c, 261