The PSI4 User's Manual

Andrew C. Simmonett Robert M. Parrish Lori A. Burns
Edward G. Hohenstein Masaaki Saitow Justin M. Turney
Rollin A. King T. Daniel Crawford C. David Sherrill

Created on: March 4, 2012

PSI4 Version: 4.0.0-beta1
Report bugs to: psicode@users.sourceforge.net

Contents

1 Introduction

1.1 Overview

`PSI4` provides a wide variety of quantum chemical methods using state-of-the-art numerical methods and algorithms. Several parts of the code feature shared-memory parallelization to run efficiently on multi-core machines (see Section 3.9. An advanced parser written in Python allows the user input to have a very simple style for routine computations, but it can also automate very complex tasks with ease.

In this section, we provide an overview of some of the features of `PSI4` along with the prerequisite steps for running calculations. Section 2 provides a brief tutorial to help new users get started. Section 3 offers further details into the structure of `PSI4` input files, and how Python can be mixed with quantum chemistry directives in `PSI4`. Section **??** provides more detail on the Python functions provided by `PSI4` and discusses some of the higher-level functions such as counterpoise correction, complete-basis-set extrapolation, and running computations on an entire database of molecules at a time. Later sections deal with the different types of computations which can be done using `PSI4` (e.g., Hartree–Fock, MP2, coupled-cluster) and general procedures such as geometry optimization and vibrational frequency analysis. The Appendix includes a complete description of all possible input keywords for each module, as well as tables of available basis sets and a listing of the sample input files available under `psi4/samples`. The user is urged to examine this directory of sample inputs, as most common types of computations are represented there. For the latest `PSI4` documentation, check `www.psicode.org`.

1.2 Citing `PSI4`

The following citation should be used in any publication utilizing the `PSI4` program package:

> J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer, R. A. King, E. F. Valeev, C. D. Sherrill, and T. D. Crawford, *WIREs: Comput. Mol. Sci.*, in press (doi: 10.1002/wcms.93).

Depending on the particular modules used, the user may also wish to cite some of the following references for theoretical, algorithmic, or implementation contributions specific to `PSI4` (in addition to appropriate references for the underlying theory):

DCFT

1. "Density Cumulant Functional Theory: First Implementation and Benchmark Results for the DCFT-06 Model," A. C. Simmonett, J. J. Wilke, H. F. Schaefer, and W. Kutzelnigg, *J. Chem. Phys.* **133**, 174122 (2010).

ADC(2)

1. "Intermediate state representation approach to physical properties of electronically excited molecules," J. Schirmer, and A. B. Trofimov, *J. Chem. Phys.* **120**, 11449-11464 (2004).

PR-CIS(D) and PR-ADC(2)

1. "Excited State Calculation for Free-Base and Metalloporphyrins with the Partially Renormalized Polarization Propagator Approach," M. Saitow and Y. Mochizuki, *Chem. Phys. Lett.*, in press.

CI

1. "The Configuration Interaction Method: Advances in Highly Correlated Approaches," C. D. Sherrill and H. F. Schaefer, in *Adv. Quantum Chem.*, vol. 34, P.-O. Löwdin, Ed. (Academic Press, New York, 1999), pp. 143-269.

CC

1. "An Introduction to Coupled Cluster Theory for Computational Chemists," T. D. Crawford and H. F. Schaefer, *Rev. Comp. Chem.* **14**, 33-136 (2000).

Mk-MRCCSD

1. "Coupling Term Derivation and General Implementation of State-Specific Multireference Coupled-Cluster Theories," F. A. Evangelista, W. D. Allen, and H. F. Schaefer, *J. Chem. Phys.* **127**, 024102 (2007).

Mk-MRCCSDT(-n)

1. "Triple Excitations in State-Specific Multireference Coupled Cluster Theory: Application of Mk-MRCCSDT and Mk-MRCCSDT-n Methods to Model Systems," F. A. Evangelista, A. C. Simmonett, W. D. Allen, H. F. Schaefer, and J. Gauss, *J. Chem. Phys.* **128**, 124104 (2008).

SAPT (General)

All capabilities of the `sapt` module are based on Symmetry Adapted Perturbation Theory. A good review article for this method is as follows:

1. "Perturbation Theory Approach to Intermolecular Potential Energy Surfaces of van der Waals Complexes," B. Jeziorski, R. Moszynski, and K. Szalewicz, *Chem. Rev.* **94**, 1887-1930 (1994).

The particular implementation and algorithms for various orders of SAPT available in `PSI4` are provided below.

SAPT0

1. "Large-scale Symmetry-adapted Perturbation Theory Computations via Density Fitting and Laplace Transformation Techniques: Investigating the Fundamental Forces of DNA-Intercalator Interactions," E. G. Hohenstein, R. M. Parrish, C. D. Sherrill, J. M. Turney, and H. F. Schaefer, *J. Chem. Phys.* **135**, 174017 (2011).
2. "Density Fitting and Cholesky Decomposition Approximations in Symmetry-Adapted Perturbation Theory: Implementation and Application to Probe the Nature of π-π Interactions in Linear Acenes," E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.* **132**, 184111 (2010).

SAPT2, SAPT2+, SAPT2+(3), SAPT2+3

1. "Density Fitting of Intramonomer Correlation Effects in Symmetry-Adapted Perturbation Theory," E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.* **133**, 014101 (2010).
2. "Wavefunction Methods for Noncovalent Interactions," E. G. Hohenstein and C. D. Sherrill, *WIREs: Comput. Mol. Sci.*, in press.

Using Natural Orbitals in SAPT

1. "Efficient Evaluation of Triple Excitations in Symmetry-Adapted Perturbation Theory via MP2 Natural Orbitals," E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.* **133**, 104107 (2010).

1.3 Obtaining and Installing `PSI4`

The latest version of the `PSI4` program package may be obtained at `www.psicode.org`. The source code is available as a gzipped tar archive (named, for example, `psi4.X.tar.gz`), and binaries may be available for certain architectures. For detailed installation and testing instructions, please refer to the installation instructions at the `PSI4` website above or to the file `psi4/INSTALL` distributed with the package.

1.3.1 Scratch File Configuration

One very important part of user configuration at the end of the installation process is to tell `PSI4` where to write its temporary ("scratch") files. Electronic structure packages like `PSI4` can create rather large temporary disk files. It is very important to ensure that PSI4 is writing its temporary files to a disk drive phsyically attached to the computer running the computation. If it is not, it will significantly slow down the program and the network. By default, PSI4 will write temporary files to `/tmp`, but this directory is often not large enough for typical computations. Therefore, you need to (a) make sure there is a sufficiently large directory on a locally-attached disk drive (100GB–1TB or more, depending on the size of the molecules to be studied) and (b) tell `PSI4` the path to this directory. The `PSI4` installation instructions explain how to set up a resource file, `.psi4rc`, for each user providing this information.

1.4 Supported Architectures

The majority of `PSI4` was developed on Mac and Linux machines. In principle, it should work on any Unix system; however, we have not tested extensively on systems other than Mac and Linux. There is not a Windows version of `PSI4`.

`PSI4` has been successfully compiled using Intel, GCC, and Clang compilers. For the Intel compilers, use versions 11 or 12.1 (we have had trouble with version 12.0).

1.5 Capabilities

`PSI4` can perform *ab initio* computations employing basis sets of contrated Gaussian-type functions of virtually arbitrary orbital quantum number. Many parts of `PSI4` can recognize and exploit the largest Abelian subgroup of the molecular point group. Table 1 displays the range of theoretical methods available in `PSI4`.

Geometry optimization (currently restricted to true minima on the potential energy surface) can be performed using either analytic gradients or energy points. Likewise, vibrational frequencies can be computed by analytic second derivatives, by finite differences of analytic gradients, or by finite differences of energies. `PSI4` can also compute an extensive list of one-electron properties.

1.6 Technical Support

The `PSI4` package is distributed for free and without any guarantee of reliability, accuracy, or suitability for any particular purpose. No obligation to provide technical support is expressed or implied. As time allows, the developers will attempt to answer inquiries directed to `crawdad@vt.edu`. For bug reports, specific and detailed information, with example inputs, would be appreciated. Questions or comments regarding this user's manual may be sent to `sherrill@gatech.edu`.

Table 1: Summary of theoretical methods available in PSI4.

Method	Energy	Gradient
RHF/ROHF/UHF SCF	Y	N
RHF/ROHF/UHF DF-SCF	Y	N
CIS/RPA/TDHF	Y	N
UHF DCFT	Y	N
RHF SAPT	Y	N
RHF MP2	Y	Y
UHF/ROHF MP2	Y	N
RHF DF-MP2	Y	N
RHF ADC(2)	Y	N
RHF/ROHF CI(n)	Y	N
RHF/ROHF RAS-CI	Y	N
RHF/ROHF MP(n)	Y	N
RHF/ROHF ZAPT(n)	Y	N
RHF/UHF/ROHF CC2	Y	N
RHF/UHF/ROHF CCSD	Y	Y
RHF/UHF/ROHF CCSD(T)	Y	Y*
RHF/UHF/ROHF EOM-CCSD	Y	Y
RHF/UHF/ROHF CC3	Y	N

* UHF-CCSD(T) gradients only, as of 4.0.0-beta1

2 A PSI4 Tutorial

2.1 Basic Input File Structure

PSI4 reads input from a text file, which can be prepared in any standard text editor. The default input file name is `input.dat` and the default output file name is `output.dat`. So that you can give your files meaningful names, these defaults can be changed by specifying the input file name and output file name on the the command line. The syntax is:

```
psi4 input-name output-name
```

2.2 Running a Basic Hartree–Fock Calculation

In our first example, we will consider a Hartree–Fock SCF computation for the water molecule using a cc-pVDZ basis set. We will specify the geometry of our water molecule using a standard Z-matrix.

```
# Any line starting with the # character is a comment line
# Here's a sample HF/cc-pVDZ H2O computation

molecule h2o {
  O
  H 1 0.96
  H 1 0.96 2 104.5
}

set basis cc-pVDZ
energy('scf')
```

For your convenience, this example can be found in `psi4/samples/tu1-h2o-energy/input.dat`. You can run it if you wish. Once `PSI4` is in your path (see the User Configuration section of the installation instructions), you can run this computation by typing

```
psi4 input.dat output.dat
```

If everything goes well, the computation should complete and should report a final restricted Hartree–Fock energy in a section like this:

```
  Energy converged.

  @RHF Final Energy:   -76.02665366589162
```

By default, the energy should be converged to about 1.0×10^{-8}, so agreement is only expected for about the first 8 digits after the decimal. If the computation does not complete, there is probably a problem with the compilation or installation of the program (see the installation instructions in Sec. 1.3).

This very simple input is sufficient to run the requested information. Notice that we didn't tell the program some otherwise useful information like the charge on the molecule (0, it's neutral), the spin multiplicity (1 for a closed-shell molecule with all electrons paired), or the reference wavefunction to use (restricted Hartree–Fock, or RHF, is usually appropriate for a closed-shell molecule). The program correctly guessed all of these options for us. We can change the default behavior through additional keywords.

Let's consider what we would do for an open-shell molecule, where not all electrons are paired. For example, let's run a computation on methylene (CH_2), whose ground electronic state has two unpaired electrons (triplet electronic state, or a spin multiplicity $2S + 1 = 3$). In this case, the default spin multiplicity (1) is not correct, so we need to tell the program the true value (3). Like many programs, `PSI4` can get the charge and multiplicity as the first two integers in the Z-matrix. Note the line with `0 3` at the beginning of the molecule specification below. In this example we will also specify the bond length and bond angle as variables (`R` and `A`), whose values are given at the end of the Z-matrix specification.

```
# Here's a sample UHF/6-31G** CH2 computation

molecule ch2 {
  0 3
  C
  H 1 R
  H 1 R 2 A

  R = 1.075
  A = 133.93
}

set basis 6-31G**
set reference uhf

energy ('scf')
```

This sample input can be found in `psi4/samples/tu2-ch2-energy`, and as before it can be run through the command `psi4 input.dat output.dat` (actually, because `psi4` by default looks for an input file named `input.dat` and writes by default to a file called `output.dat`, in this case one could also just type `psi4`). If it works, it should print the final energy as

```
  @UHF Final Energy:   -38.92534160932308
```

Notice we added a new keyword, `set reference uhf`, to the input. For open-shell molecules, we have a choice of unrestricted orbitals (unrestricted Hartree–Fock, or UHF), or restricted orbitals (restricted open-shell Hartree–Fock, or ROHF). Usually, UHF is a little easier to converge (although it may be more susceptible to spin contamination than ROHF).

2.3 Geometry Optimization and Vibrational Frequency Analysis

The above examples were simple single-point energy computations (as specified by the `energy()` function). Of course there are other kinds of computations to perform, such as geometry optimizations and vibrational frequency computations. These can be specified by replacing `energy()` with `optimize()` or `frequencies()`, respectively.

Here's an example of optimizing the H_2O molecule using Hartree–Fock with a cc-pVDZ basis set (located in `psi4/samples/tu3-h2o-opt`).

```
# Optimize H2O HF/cc-pVDZ

molecule h2o {
  O
  H 1 0.96
  H 1 0.96 2 104.5
}

set basis cc-pVDZ
optimize('scf')
```

This should perform a series of gradient computations. The gradient points which way is downhill in energy, and the optimizer then modifies the geometry to follow the gradient. After about 4 cycles, the geometry should converge with a message like `Optimization is complete!`. As indicated in the following table (printed by the optimizer at the end of the computation), the energy decreases with each step, and the maximum force on each atom also decreases with each step (in principle these numbers could increase in some iterations, but here they do not).

```
----------------------------------------------------------------------
Step         Energy             Delta(E)      MAX force   MAX Delta(q)
----------------------------------------------------------------------
  1    -76.026653665892    -76.026653665892    1.52e-02   1.52e-02
  2    -76.026907793199     -0.000254127307    9.55e-03   9.55e-03
  3    -76.027052927171     -0.000145133972    4.47e-04   4.47e-04
  4    -76.027053472137     -0.000000544965    1.16e-04   1.16e-04
----------------------------------------------------------------------
```

To get harmonic vibrational frequencies, *first we must set up an input using the OPTIMIZED GEOMETRY*. We can easily get the optimized geometry from the previous computation. Looking at the output from running the previous example, we see that the OH bond length is about 0.9463 Ångstroms, and the bond angle is about 104.575 degrees. It's good to give this many digits (or more) to make sure there's not significant roundoff error in the geometry when running a frequency computation. So, our frequency computation input (which can be found as test case `psi4/samples/tu4-h2o-freq`) is:

```
# Frequencies for H2O HF/cc-pVDZ at optimized geometry

molecule h2o {
  O
  H 1 0.9463
  H 1 0.9463 2 104.575
}

set basis cc-pVDZ
frequencies('scf')
```

Alternatively, it's also possible for `PSI4` to use Cartesian coordinate input. Here, the Cartesian coordinates of the optimized geometry can be extracted from the *bottom* of the optimization output. The input would then look like this:

```
molecule h2o {
  O     0.0000000000  -0.0000000000  -0.1224239500
  H     0.0000000000  -1.4147069876   0.9714784639
  H    -0.0000000000   1.4147069876   0.9714784639
}

set basis cc-pVDZ
frequencies('scf')
```

If either of the inputs above are run, the program should do some computations and then finally report the following harmonic vibrational frequencies (roundoff errors of around 0.1 cm^{-1} may exist):

```
          Irrep      Harmonic Frequency
                          (cm-1)
        -----------------------------------------------
             A1         1776.1735
             A1         4113.8031
          B2         4211.7879
        -----------------------------------------------
```

Notice that the symmetry type of the normal modes is specified (A1, A1, B2). The program also prints out the normal modes in terms of Cartesian coordinates of each atom. For example, the normal mode at 1776 cm^{-1} is

```
   Frequency:       1776.17
   Force constant:   0.1194
            X       Y       Z
  O       0.000   0.000  -0.067
  H       0.000   0.416   0.536
  H       0.000  -0.416   0.536
```

where the table shows the displacements in the X, Y, and Z dimensions for each atom along the normal mode coordinate. (This information could be used to animate the vibrational frequency using visualization software.)

2.4 Analysis of Intermolecular Interactions

Now let's consider something a little more interesting. PSI4 contains code to analyze the nature of intermolecular interactions between two molecules, via symmetry-adapted perturbation theory (SAPT) [1]. This kind of analysis gives a lot of insight into the nature of intermolecular interactions, and PSI4 makes these computations easier than ever.

For a SAPT computation, the input needs to provide information on two distinct molecules. This is very easy, we just give a Z-matrix or set of Cartesian coordinates for each molecule, and separate the two with two dashes, like this:

```
# Example SAPT computation for ethene*ethine (i.e., ethylene*acetylene),
# test case 16 from the S22 database

molecule dimer {
0 1
C   0.000000  -0.667578  -2.124659
C   0.000000   0.667578  -2.124659
H   0.923621  -1.232253  -2.126185
H  -0.923621  -1.232253  -2.126185
H  -0.923621   1.232253  -2.126185
H   0.923621   1.232253  -2.126185
--
0 1
C   0.000000   0.000000   2.900503
C   0.000000   0.000000   1.693240
H   0.000000   0.000000   0.627352
H   0.000000   0.000000   3.963929
units angstrom

no_reorient
symmetry c1
}
```

Notice we have a couple of new keywords in the molecule specification. `no_reorient` tells the program not to reorient the molecule to a different coordinate system (this is important for SAPT to make sure the same coordinate frame is used for the computations of the two monomers and for the dimer). The next keyword tells `PSI4` to run in C1 point-group symmetry (i.e., without using symmetry), even if a higher symmetry is detected. SAPT computations need to be run with symmetry turned off in this way.

Here's the second half of the input:

```
set globals {
    basis jun-cc-pVDZ
    scf_type DF
    freeze_core True
}

energy('sapt0')
```

Before, we have been setting keywords individually with commands like `set basis cc-pVDZ`. Because we have a few more options now, it's convenient to place them together into the `set globals` or `set` block, bounded by `{...}`. This will set all of these options as "global" options (meaning that they are visible to all parts of the program). Most common `PSI4` options can be set in a `globals` section like this. If an option needs to be visible only to one part of the program (e.g., we only want to increase the energy convergence in the SCF code, but not the rest of the code), it can be placed in a section of input visible to that part of the program (e.g., `set scf e_convergence 1.0E-8`).

Back to our SAPT example, we have found that for basic-level SAPT computations (i.e., SAPT0), a good error cancellation is found [2] with the jun-cc-pVDZ basis (this is the usual aug-cc-pVDZ basis, but without diffuse functions on hydrogen and without diffuse d functions on heavy atoms) [3]. So, we'll use that as our standard basis set. The SAPT code is designed to use density fitting techniques, because they introduce minimal errors whil providing much faster computations [4, 5]. Since we're using density fitting for the SAPT, we might as well also use it for the Hartree–Fock computations that are performed as part of the SAPT. We can specify that with the option and value pair `SCF_TYPE` `DF`.

Density fitting procedures require the use of auxiliary basis sets that pair with the primary basis set. Fortunately, `PSI4` is usually smart enough to figure out what auxiliary basis sets are needed for a given computation. In this case, jun-cc-pVDZ is a standard enough basis set (just a simple truncation of the very popular aug-cc-pVDZ basis set) that `PSI4` correctly guesses that we want the jun-cc-pVDZ-JKFIT auxiliary basis for the Hartree–Fock, and the jun-cc-pVDZ-RI basis set for the SAPT procedure.

To speed up the computation a little, we also tell the SAPT procedure to freeze the core electrons with the `FREEZE_CORE` option. The SAPT procedure is invoked with the simple call, `energy('sapt0')`. This call knows to automatically run two monomer computations and a dimer computation and then use these results to perform the SAPT analysis. The various energy components are printed at the end of the output, in addition to the total SAPT0 interaction energy. An explanation of the various energy components can be found in the review by Jeziorski, Moszynski, and Szalewicz [1], and this is discussed in more detail in the SAPT section later in this manual. For now, we'll note that most of the SAPT energy components are negative; this means those are attractive contributions (the zero of energy in a SAPT computation is defined as non-interacting monomers). The exchange contributions are positive (repulsive). In this example, the most attractive contribution between ethylene and acetylene is an electrostatic term of -2.12 kcal mol^{-1} (`Elst10,r` where the 1 indicates the first-order perturbation theory result with respect to the intermolecular interaction, and the 0 indicates zeroth-order with respect to intramolecular electron correlation). The next most attractive contribution is the `Disp20` term (2nd order intermolecular dispersion, which looks like an MP2 in which one excitation is placed on each monomer), contributing an attraction of -1.21 kcal mol^{-1}. It is not surprising that the electrostatic contribution is dominant, because the geometry chosen for this example has the acetylene perpendicular to the ethylene, with the acetylene hydrogen pointing directly at the double bond in ethylene; this will be attractive because the H atoms in acetylene bear a partial positive charge, while the electron-rich double bond in ethylene bears a partial negative charge. At the same time, the dispersion interaction should be smaller because the perpendicular geometry does not allow much overlap between the monomers. Hence, the SAPT analysis helps clarify (and quantify) our physical understanding about the interaction between the two monomers.

2.5 Potential Surface Scans and Counterpoise Correction Made Easy with Psithon

Finally, let's consider an example that shows how the Python driver in `PSI4` simplifies some routine tasks. `PSI4` can interpret valid Python code in addition to the computational chemistry directives we've seen in the previous examples; we call this mixture Psithon. The Python computer language is very easy to pick up, and even users previously unfamiliar with Python can use it to simplify tasks by modifying some of the example input files supplied with `PSI4` in the `psi4/samples` directory.

Suppose you want to do a limited potential energy surface scan, such as computing the interaction energy between two neon atoms at various interatomic distances. One simple but unappealing way to do this is to create separate input files for each distance to be studied. But most of these input files are identical, except that the interatomic distance is different. Psithon lets you specify all this in a single input file, looping over the different distances with an array like this: `Rvals=[2.5, 3.0, 4.0]`.

Let's also suppose you want to do counterpoise (CP) corrected energies. Counterpoise correction involves computing the dimer energy and then subtracting out the energies of the two monomers, each evaluated in the dimer basis. Again, each of these computations could be run in a separate input file, but because counterpoise correction is a fairly standard procedure for intermolecular interactions, `PSI4` knows about it and has a built-in routine to perform counterpoise correction. It only needs to know what method you want to do the counterpoise correction on (here, let's consider CCSD(T)), and it needs to know what's monomer A and what's monomer B. This last issue of specifying the monomers separately was already dealt with in the previous SAPT example, where we saw that two dashes in the `molecule` block can be used to separate monomers.

So, we're going to do counterpoise-corrected CCSD(T) energies for Ne_2 at a series of different interatomic distances. And let's print out a table of the interatomic distances we've considered, and the CP-corrected CCSD(T) interaction energies (in kcal mol^{-1}) at each geometry. Doing all this in a single input is surprisingly easy in `PSI4`. Here's the input (available as `psi4/samples/tu6-cp-ne2`).

```
#! Example potential energy surface scan and CP-correction for Ne2

molecule dimer {
  Ne
--
  Ne 1 R
}

Rvals=[2.5, 3.0, 4.0]

set basis aug-cc-pVDZ
set freeze_core True

# Initialize a blank dictionary of counterpoise corrected energies
# (Need this for the syntax below to work)
ecp = {}

for R in Rvals:
  dimer.R = R
  ecp[R] = cp('ccsd(t)')

print "\n"
print "CP-corrected CCSD(T)/aug-cc-pVDZ interaction energies\n\n"
print "        R (Ang)         E_int (kcal/mol)            \n"
print "-----------------------------------------------------\n"
for R in Rvals:
  e = ecp[R] * psi_hartree2kcalmol
  print "       %3.1f           %10.6f\n" % (R, e)
```

First, you can see the `molecule` block has a couple dashes to separate the monomers from each other. Also note we've used a Z-matrix to specify the geometry, and we've used a variable (`R`) as the interatomic distance. We have *not* specified the value of `R` in the `molecule` block like we normally would. That's because we're going to vary it during the scan across the potential energy surface. Below the `molecule` block, you can see the `Rvals` array specified. This is a Python array holding the interatomic distances we want to consider. In Python, arrays are surrounded by square brackets, and elements are separated by commas.

The next lines, `set basis aug-cc-pVDZ` and `set freeze_core True`, are familiar from previous test cases. Next comes a slightly unusual-looking line, `ecp = {}`. This is Python's way of initializing a "dictionary." We're going to use this dictionary to store the counterpoise-corrected energies as they become available. A dictionary is like an array, but we can index it using strings or floating-point numbers instead of integers if we want. Here, we will index it using floating-point numbers, namely, the `R` values. This winds up being slightly more elegant than a regular array in later parts of the input file.

The next section, beginning with `for R in Rvals:`, loops over the interatomic distances, `R`, in our aray `Rvals`. In Python, loops need to be indented, and each line in the loop has to be indented by the same amount. The first line in the loop, `dimer.R = R`, sets the Z-matrix variable `R` of the molecule called `dimer` to the `R` value extracted from the `Rvals` array. The next line, `ecp[R] = cp('ccsd(t)')`, computes the counterpoise-corrected CCSD(T) energy and places it in the `ecp` dictionary with `R` as the index. Note we didn't need to specify ghost atoms, and we didn't need to call the monomer and dimer computations separately. The built-in Psithon function `cp()` does it all for us, automatically.

And that's it! The only remaining part of the example input is a little table of the different R values and the CP-corrected CCSD(T) energies, converted from atomic units (hartree) to kcal mol^{-1} by multiplying by the automatically-defined conversion factor `psi_hartree2kcalmol`, which is defined in section 3.1. Notice the loop over `R` to create the table looks just like the loop over `R` to run the different computations, and the CP-corrected energies `ecp[R]` are accessed the same way they were stored. The `print` line at the end just specifies some formatting for the printed table (first element is a floating point number 3 spaces wide with one digit after the decimal, and the second element is a floating point number 10 spaces wide with 6 digits after the decimal); the format strings are the same as in the C programming language. For tables more complicated than the simple one used here, Psithon has built-in support for tables (see the next section).

The following section goes over Psithon in much more detail, but hopefully this example already makes it clear that many complex tasks can be done very easily in `PSI4`.

3 Psithon

To allow arbitrarily complex computations to be performed, PSI4 was built upon the Python interpreter. However, to make the input syntax simpler, some pre-processing of the input file is performed before it is interpreted, resulting in Python syntax that is customized for PSI, termed Psithon. In this section we will describe the essential features of the Psithon language. PSI4 is distributed with an extensive test suite, described in section E; the input files for these test cases can be found in the samples subdirectory of the top-level PSI4 source directory, and should serve as useful examples.

3.1 Scratch Files, Default Variables and the .psi4rc File

For convenience, the Python interpreter will execute the contents of the .psi4rc file in the current user's home area (if present) before performing any tasks in the input file. This allows frequently used python variables to be automatically defined in all input files. For example, if we repeatedly make use of the universal gravitational constant, the following line could be placed in the .psi4rc file

```
UGC = 6.67384E-11 # m^3 / kg^-1 s^-2
```

which would make the variable UGC available in all PSI4 input files. For convenience, the physical constants used within the PSI4 code (which are obtained from the 3rd edition of the IUPAC Green book[6]) are also automatically loaded as Psithon variables (before .psi4rc is loaded, so that .psi4rc values can be overridden by the user); table 2 shows these variables. The "psi_" prefix is to prevent clashes with user-defined variables in PSI4 input files.

Another important use of the .psi4rc file is to control the handling of scratch files. PSI4 has a number of utilities that manage input and output (I/O) of quantities to and from the hard disk. Most quantities, such as molecular integrals, are intermediates that are not of interest to the user and can be deleted after the computation finishes, but pertinent details of computations are also written to a checkpoint file and might be useful in subsequent computations. All files are sequentially numbered and are written to /tmp, then deleted at the end of the computation, unless otherwise instructed by the user.

A Python callable handle to the PSI4 I/O management routines is available, and is called psi4_io. To instruct the I/O manager to send all files to another location, say /scratch/user, add the following command to the .psi4rc file (note the trailing "/"):

```
psi4_io.set_default_path('/scratch/user/')
```

For batch jobs running through a queue, it might be more convenient to use an environmental variable (in this case $MYSCRATCH) to set the scratch directory; the following code will do that:

```
scratch_dir = os.environ.get('MYSCRATCH')
```

Table 2: The physical constants used within PSI4, which are automatically made available within all PSI4 input files.

Name	Value	Description
psi_h	6.62606896E-34	The Planck constant (Js)
psi_c	2.99792458E8	Speed of light (ms^{-1})
psi_kb	1.3806504E-23	The Boltzmann constant (JK^{-1})
psi_R	8.314472	Universal gas constant ($JK^{-1}mol^{-1}$)
psi_bohr2angstroms	0.52917720859	Bohr to Angstroms conversion factor
psi_bohr2m	0.52917720859E-10	Bohr to meters conversion factor
psi_bohr2cm	0.52917720859E-8	Bohr to centimeters conversion factor
psi_amu2g	1.660538782E-24	Atomic mass units to grams conversion factor
psi_amu2kg	1.660538782E-27	Atomic mass units to kg conversion factor
psi_au2amu	5.485799097E-4	Atomic units (m_e) to atomic mass units conversion factor
psi_hartree2J	4.359744E-18	Hartree to joule conversion factor
psi_hartree2aJ	4.359744	Hartree to attojoule (10^{-18}J) conversion factor
psi_cal2J	4.184	Calorie to joule conversion factor
psi_dipmom_au2si	8.47835281E-30	Atomic units to SI units (Cm) conversion factor for dipoles
psi_dipmom_au2debye	2.54174623	Atomic units to Debye conversion factor for dipoles
psi_dipmom_debye2si	3.335640952E-30	Debye to SI units (Cm) conversion factor for dipoles
psi_c_au	137.035999679	Speed of light in atomic units
psi_hartree2ev	27.21138	Hartree to eV conversion factor
psi_hartree2wavenumbers	219474.6	Hartree to cm^{-1} conversion factor
psi_hartree2kcalmol	627.5095	Hartree to kcal mol^{-1} conversion factor
psi_hartree2MHz	6.579684E9	Hartree to MHz conversion factor
psi_kcalmol2wavenumbers	349.7551	kcal mol^{-1} to cm^{-1} conversion factor
psi_e0	8.854187817E-12	Vacuum permittivity (Fm^{-1})
psi_na	6.02214179E23	Avagadro's number
psi_me	9.10938215E-31	Electron rest mass (in kg)

```
if scratch_dir:
    psi4_io.set_default_path(scratch_dir + '/')
```

Individual files can be send to specific locations. For example, file 32 is the checkpoint file that the user might want to retain in the working directory (*i.e.*, where PSI4 was launched from) for restart purposes. This is accomplished by the commands below:

```
psi4_io.set_specific_path(32, './')
psi4_io.set_specific_retention(32, True)
```

To circumvent difficulties with running multiple jobs in the same scratch, the process ID (PID) of the PSI4 instance is incorporated into the full file name; therefore, it is safe to use the same scratch directory for calculations running simultaneously.

To override any of these defaults for selected jobs, simply place the appropriate commands from the snippets above in the input file itself. During excecution, the .psi4rc defaults will be loaded in first, but then the commands in the input file will be executed. Executing PSI4 with the -m (for messy) flag will prevent files being deleted at the end of the run:

```
psi4 -m
```

3.2 Molecule Specification

PSI4 has a very flexible input parser that allows the user to provide geometries as Cartesian coordinates, Z-matrix variables, or a combination of both. The use of fixed values and variables are supported for both. For example, the geometry for H_2 can be specified a number of ways, using the molecule keyword:

```
molecule{
  H
  H 1 0.9
}

 or

molecule{
  H
  H 1 r
  r = 0.9
}

 or

molecule{
  H1
  H2 H1 0.9
}

 or

molecule{
  H 0.0 0.0 0.0
  H 0.0 0.0 0.9
}

 or

molecule{
  H 0.0 0.0 0.0
  H 0.0 0.0 r
  r = 0.9
}

 or

molecule{
  H 0.0 0.0 -r
  H 0.0 0.0 r
  r = 0.45
}
```

Blank lines are ignored and, unlike regular Python syntax, indentation within the `molecule` block does not matter, although the `molecule` keyword itself must be aligned within the input according to standard Python syntax. For more examples of geometry specification, see the `mints1` input file in the samples folder. It is also possible to mix Cartesian and Z-matrix geometry specifications, as demonstrated in the `mints4` sample input file.

3.2.1 Multiple Molecules

To facilitate more elaborate computations, it is possible to provide a name for each molecule, and tell `PSI4` which one should be used in a given calculation. For example, consider the following input file:

```
molecule h2{
  H
  H 1 0.9
}


set basis cc-pvdz
set reference rhf
energy('scf')

molecule h{
  H
}

set basis cc-pvdz
set reference uhf
energy('scf')
```

Here, two separate jobs are performed on two different molecules; the first is performed on H_2, while the second is for H atom. The last molecule to be specified is the "active" molecule by default. To explicitly activate a named molecule, the activate keyword is provided. Using this keyword, the above input file can be equivalently written as follows:

```
molecule h2{
  H
  H 1 0.9
}

molecule h{
  H
}

activate(h2)
set basis cc-pvdz
set reference rhf
energy('scf')

activate(h)
set basis cc-pvdz
set reference uhf
energy('scf')
```

Note that whenever the molecule is changed, the basis set must be specified again. The following section provides more details about the job control keywords used in the above examples.

3.2.2 Additional Keywords

In addition to specifying the geometry, additional information can be provided in the `molecule` block. If two integers are encountered on any line of the `molecule` block, they are interpreted as the molecular charge and multiplicity ($2 \times M_s + 1$), respectively. The symmetry can be specified by a line reading `symmetry` ***`symbol`***, where ***`symbol`*** is the Schönflies symbol of the (Abelian) point group to use for the computation; see section 3.4 for more details. This need not be specified, as the molecular symmetry is automatically detected by `PSI4`. Certain computations require that the molecule is not reoriented; this can be achieved by adding either `no_reorient` or `noreorient`. By default, Ångström units are used; this is changed by adding a line that reads `units` ***`spec`***, where ***`spec`*** is one of `ang`, `angstrom`, `a.u.`, `au`, or `bohr`.

3.3 Geometries from the PubChem Database

Obtaining rough starting guess geometries can be burdensome. The Z-matrix coordinate system was designed to provide chemists with an intuitive method for guessing structures in terms of bond lengths and angles. While Z-matrix input is intuitive for small molecules with few degrees of freedom, it quickly becomes laborious as the system size grows. To obtain a reasonable starting guess geometry, PSI4 can take a chemical name as input; this is then used to attempt to retrieve Cartesian coordinates from the PubChem database.[7]

For example, to run a computation on benzene, we can use the following molecule specification:

```
molecule benzene {
    pubchem:benzene
}
```

If the computer is connected to the internet, the above code will instruct PSI4 to search PubChem for a starting structure. The search is actually performed for compounds whose name *contains* "benzene", so multiple entries will be returned. If the name provided ("benzene" in the above example) exactly matches one of the results, that entry will be used. If no exact match is found the results, along with a unique chemical identifier (CID), are printed to the output file, prompting the user to provide a more specific name. For example, if we know that we want to run a computation on a compound whose name(s) contain "benzene", but we're not sure of the exact IUPAC name, the following input can be used:

```
molecule benzene {
    pubchem:benzene*
}
```

Appending the "*" prevents an exact match from being found and, at the time of writing, the following results are displayed in the output file:

```
     Chemical ID     IUPAC Name
             241   benzene
            7371   benzenesulfonic acid
           91526   benzenesulfonate
             244   phenylmethanol
             727   1,2,3,4,5,6-hexachlorocyclohexane
             240   benzaldehyde
           65723   benzenesulfonohydrazide
           74296   N-phenylbenzenesulfonamide
             289   benzene-1,2-diol
             243   benzoic acid
```

```
      7370   benzenesulfonamide
    636822   1,2,4-trimethoxy-5-[(E)-prop-1-enyl]benzene
      7369   benzenesulfonyl chloride
     12932   N-[2-di(propan-2-yloxy)phosphinothioylsulfanylethyl]benzenesulfonamide
      7505   benzonitrile
     78438   N-[anilino(phenyl)phosphoryl]aniline
     12581   3-phenylpropanenitrile
    517327   sodium benzenesulfonate
    637563   1-methoxy-4-[(E)-prop-1-enyl]benzene
    252325   [(E)-prop-1-enyl]benzene
```

Note that some of these results do not contain the string “benzene”; these compounds have synonyms containing that text. We can now replace the “benzene*” in the input file with one of the above compounds using either the IUPAC name or the CID provided in the list, *viz*:

```
molecule benzene {
    pubchem:637563
}

 or

molecule benzene {
    pubchem:1-methoxy-4-[(E)-prop-1-enyl]benzene
}
```

Some of the structures in the database are quite loosely optimized and do not have the correct symmetry. Before starting the computation, PSI4 will check to see if the molecule is close to having each of the possible symmetries, and will adjust the structure accordingly so that the maximum symmetry is utilized.

The standard keywords, described in section 3.2.2, can be used in conjuction to specify charge, multiplicity, symmetry to use, *etc.* .

3.4 Symmetry

For efficiency, PSI4 can utilize the largest Abelian subgroup of the full point group of the molecule. Concomitantly a number of quantities, such as SOCC and DOCC, are arrays whose entries pertain to irreducible representations (irreps) of the molecular point group. Ordering of irreps follows the convention used in Cotton’s *Chemical Applications of Group Theory*, as detailed in Table 3. We refer to this convention as “Cotton Ordering” hereafter.

For example, water ($C_{2\text{v}}$ symmetry) has 3 doubly occupied A_1 orbitals, as well as 1 each of B_1 and B_2 symmetry; the corresponding DOCC array is therefore:

Table 3: The ordering of irreducible representations (irreps) used in PSI4.

Point Group	Irrep Order							
C_1	A							
C_i	A_g	A_u						
C_2	A	B						
C_s	A'	A''						
D_2	A	B_1	B_2	B_3				
C_2v	A_1	A_2	B_1	B_2				
C_2h	A_g	B_g	A_u	B_u				
D_2h	A_g	B_1g	B_2g	B_3g	A_u	B_1u	B_2u	B_3u

```
  DOCC = [3, 0, 1, 1]
```

Although PSI4 will detect the symmetry automatically, and use the largest possible Abelian subgroup, the user might want to run in a lower point group. To do this the symmetry keyword can be used when inputting the molecule (see section 3.2). In most cases the standard Schönflies symbol (one of c1, c2, ci, cs, d2, c2h, c2v, d2h) will suffice. For certain computations, the user might want to specify which particular subgroup is to be used by appending a unique axis specifier. For example when running a computation on a molecule with D_2h symmetry in C_2v, the C_2 axis can be chosen as either the x, the y, or the z; these can be specified by requesing the symmetry as c2vx, c2vy, or c2vz, respectively. Likewise the c2x, c2y, c2z, c2hx, c2hy, and c2hz labels are valid. For C_s symmetry the labels csx, csy, and csz request the yz, xz, and xy planes be used as the mirror plane, respectively. If no unique axis is specified, PSI4 will choose an appropriate subgroup.

3.5 Non-Covalently Bonded Molecule Fragments

PSI4 has an extensive range of tools for treating non-covalent intermolecular forces, including counterpoise corrections and symmetry adapted perturbation theory methods. These require the definition of which fragments are interacting within the complex. PSI4 provides a very simple mechanism for doing so; simply define the complex's geometry using the standard Cartesian, Z-matrix, or mixture thereof, specifications and then place two dashes between nonbonded fragements. For example, to study the interaction energy of ethane and ethyne molecules, we can use the following molecule block:

```
molecule{
  0 1
  C  0.000000 -0.667578  -2.124659
  C  0.000000  0.667578  -2.124659
  H  0.923621 -1.232253  -2.126185
  H -0.923621 -1.232253  -2.126185
  H -0.923621  1.232253  -2.126185
  H  0.923621  1.232253  -2.126185
  --
  0 1
  C 0.000000 0.000000 2.900503
  C 0.000000 0.000000 1.693240
  H 0.000000 0.000000 0.627352
  H 0.000000 0.000000 3.963929
}
```

In this case, the charge and multiplicity of each interacting fragment is explicitly specified. If the charge and multiplicity are specified for the first fragment, it is assumed to be the same for all fragments. When considering interacting fragments, the overall charge is simply the sum of all fragment charges, and any unpaired electrons are assumed to be coupled to yield the highest possible M_s value.

3.6 Job Control

PSI4 comprises a number of modules, written in C++, that each perform specific tasks and are callable directly from the Python front end. Each module recognizes specific keywords in the input file, detailed in Appendix B, which control its function. The keywords can be made global, or scoped to apply to certain specific modules. The following examples demonstrate some of the ways that global keywords can be specified:

```
set globals basis cc-pVDZ

 or

set basis cc-pVDZ

 or

set globals basis = cc-pVDZ

 or

set basis = cc-pVDZ

 or

set globals{
  basis cc-pVDZ
}
```

```
 or

set{
  basis cc-pVDZ
}

 or

set{
  basis = cc-pVDZ
}
```

Note the lack of quotes around "cc-pVDZ", even though it is a string. The Psithon preprocessor automatically wraps any string values in `set` commands in strings. The last three examples provide a more convenient way for specifying multiple keywords:

```
set{
  basis = cc-pVDZ
  print = 1
  reference = rhf
}
```

For arguments that require an array input, standard Python list syntax should be used, viz.:

```
set{
  docc = [3, 0, 1, 1]
}
```

List / matrix inputs may span multiple lines, as long as the opening "[" is on the same line as the name of the keyword.

Any of the above keyword specifications can be scoped to individual modules, by adding the name of the module after the `set` keyword. Omitting the module name, or using the name `global` or `globals` will result in the keyword being applied to all modules. For example, in the following input

```
molecule{
  o
  h 1 roh
  h 1 roh 2 ahoh
  roh = 0.957
  ahoh = 104.5
}

set basis cc-pVDZ
set ccenergy print 3
set scf print 1
energy('ccsd')
```

the basis set is set to cc-pVDZ throughout, the SCF code will have a print level of 1 and the `ccenergy` code, which performs coupled cluster computations, will use a print level of 3. In this example a full CCSD computation is performed by running the SCF code first, then the coupled cluster modules; the `energy()` Python helper function ensures that this is performed correctly. Note that the Python interpreter executes commands in the order they appear in the input file, so if the last four commands in the above example were to read

```
set basis cc-pVDZ
energy('ccsd')
set ccenergy print 3
set scf print 1
```

the commands that set the print level would be ineffective, as they would be processed after the CCSD computation completes.

3.7 Assigning Basis Sets

While the above syntax will suffice for specifying basis sets in most cases, the user may need to assign basis sets to specific atoms. To achieve this, a `basis` block can be used. We use a snippet from the `mints2` sample input file, which performs a benzene SCF computation, to demonstrate this feature.

```
basis {
   assign DZ
   assign C 3-21G
   assign H1 sto-3g
   assign C1 sto-3g
}
```

The first line in this block assigns the DZ basis set to all atoms. The next line then assigns 3-21G to all carbon atoms, leaving the hydrogens with the DZ basis set. On the third line, the hydrogen atoms which have been specifically labelled as H1 aregiven the STO-3G basis set, leaving the unlabelled hydrogen atoms with the DZ basis set. Likewise, the fourth line assigns the STO-3G basis set to just the carbon atoms labelled C1. This bizzare example was constructed to demonstrate the syntax, but the flexibility of the basis set specification is advantageous, for example, when selectivily omitting diffuse functions to make computations more tractable.

In the above example the basis sets have been assigned asymmetrically, reducing the effective symmetry from D_{6h} to C_{2v}; PSI4 will detect this automatically and run in the appropriate point group. The same syntax can be used to specify basis sets other than that used to define orbitals. For example,

```
set df_basis_mp2 cc-pvdz-ri

 or

basis {
   assign cc-pVDZ-RI df_basis_mp2
}
```

are both equivalent ways to set the auxiliary basis set for density fitted MP2 computations. To assign the aug-cc-pVDZ-RI to carbon atoms, the following command is used:

```
basis {
   assign C aug-cc-pVDZ-RI df_basis_mp2
}
```

When Dunning's correlation consistent basis sets (cc-pVXZ), and core-valence and diffuse variants thereof, are being used the SCF and DF-MP2 codes will chose the appropriate auxilliary basis set automatically, unless instructed otherwise by setting theauxiliary basis set in the input. Finally, we note that the `basis` block may also be used for defining basis sets, as detailed in section 4.2.

3.8 Memory Specification

By default, PSI4 assumes that 256 Mb of memory are available. While this is enough for many computations, many of the algorithms will perform better if more is available. To specify memory, the `memory` keyword should be used. The following lines are all equivalent methods for specifying that 2 Gb of RAM is available to PSI4:

```
memory 2 Gb
```

```
 or

memory 2000 Mb

 or

memory 2000000 Kb
```

One convenient way to override the PSI4 default memory is to place a memory command in the .psi4rc file, as detailed in section 3.1.

3.9 Threading

Most new modules in PSI4 are designed to run efficiently on SMP architectures via application of several thread models. The de facto standard for PSI4 involves using threaded BLAS/LAPACK (particularly Intel's excellent MKL package) for most tensor-like operations, OpenMP for more general operations, and Boost Threads for some special-case operations. Note: Using OpenMP alone is a really bad idea. The developers make little to no effort to explicitly parallelize operations which are already easily threaded by MKL or other threaded BLAS. Less than 20% of the threaded code in PSI4 uses OpenMP, the rest is handled by parallel DGEMM and other library routines. From this point forward, it is assumed that you have compiled PSI4 with OpenMP and MKL (Note that it is possible to use g++ or another compiler, and yet still link against MKL).

Control of threading in PSI4 can be accomplished at a variety of levels, ranging from global environment variables to direct control of thread count in the input file, to even directives specific to each model. This hierarchy is explained below. Note that each deeper level trumps all previous levels.

(1) OpenMP/MKL Environment Variables

The easiest/least visible way to thread PSI4 is to set the standard OpenMP/MKL environment variables. For instance, in tcsh:

```
setenv OMP_NUM_THREADS 4
setenv MKL_NUM_THREADS 4
```

PSI4 then detects these value via the API routines in `<omp.h>` and `<mkl.h>`, and runs all applicable code with 4 threads. These environment variables are typically defined in a .tcshrc or .bashrc.

(2) The -n Command Line Flag

To change the number of threads at runtime, the -n flag may be used. An example is:

```
psi4 -i input.dat -o output.dat -n 4
```

which will run on four threads.

(3) Setting Thread Numbers in an Input

For more explicit control, the Process::environment class in `PSI4` can override the number of threads set by environment variables. This functionality is accessed via the `set_num_threads` Psithon function, which controls both MKL and OpenMP thread numbers. The number of threads may be changed multiple times in a `PSI4` input file. An example input for this feature is:

```
# A bit small-ish, but you get the idea
molecule h2o {
0 1
O
H 1 1.0
H 1 1.0 2 90.0
}

set scf {
basis cc-pvdz
scf_type df
}

# Run from 1 to 4 threads, for instance, to record timings
for nthread in range(1,5):
    set_num_threads(nthread)
    energy('scf')
```

(4) Method-Specific Control

Even more control is possible in certain circumstances. For instance, the threaded generation of AO density-fitted integrals involves a memory requirement proportional to the number of threads. This requirement may exceed the total memory of a small-memory node if all threads are involved in the generation of these integrals. For general DF algorithms, the user may specify:

```
set MODULE_NAME df_ints_num_threads n
```

to explicitly control the number of threads used for integral formation. Setting this variable to 0 (the default) uses the number of threads specified by the `set_num_threads()` Psithon method or the default environmental variables.

3.10 Return Values and PSI Variables

To harness the power of Python, PSI4 makes the most pertinent results of each computation are made available to the Python interpreter for post-processing. To demonstrate, we can embellish the previous example of H_2 and H atom:

```
molecule h2{
  H
  H 1 0.9
}

set basis cc-pvdz
set reference rhf
h2_energy = energy('scf')

molecule h{
  H
}

set basis cc-pvdz
set reference uhf
h_energy = energy('scf')

D_e = psi_hartree2kcalmol*(2*h_energy - h2_energy)
print"De=%f"%D_e
```

The `energy()` function returns the final result of the computation, which we assign to a Python variable. The two energies are then converted to a dissociation energy and printed to the output file using standard Python notation. Sometimes there are multiple quantities of interest; these can be accessed through the `get_variable()` function. For example, after performing a density fitted MP2 computation, both the spin component scaled energy and the unscaled MP2 energy are made available:

```
e_mp2=get_variable('DF-MP2 TOTAL ENERGY')
e_scs_mp2=get_variable('SCS-DF-MP2 TOTAL ENERGY')
```

Each module and the Python driver set PSI variables over the course of a calculation. The values for all can be printed in the output file with the input file command `print_variables()` . Note that PSI variables accumulate over a PSI4 instance and are not cleared by `clean()`. So if you run in a single input file a STO-3G FCI followed by a aug-cc-pVQZ SCF followed by a `print_variables()` command, the last will include both `SCF TOTAL ENERGY` and `FCI TOTAL ENERGY`. Don't get excited that you got a high-quality calculation cheaply. Refer to Appendix D for a listing of the variables set by each module.

3.11 Loops

Python provides many control structures, which can be used within PSI4 input files. For example, to loop over three basis sets, the following code can be used:

```
basis_sets=["cc-pVDZ","cc-pVTZ","cc-pVQZ"]
for basis_set in basis_sets:
    set basis = $basis_set
    energy('scf')
```

The declaration of `basis_sets` is completely standard Python, as is the next line, which iterates over the list. However, because the Psithon preprocessor wraps strings in quotes by default, we have to tell it that `basis_sets` is a Python variable, not a string, by prefixing it with a dollar sign. The geometry specification supports delayed initialization of variable, which permits potential energy scans. As an example, we can scan both the angle and bond length in water:

```
molecule h2o{
  O
  H1 R
  H1 R2 A
}

Rvals=[0.9,1.0,1.1]
Avals=range(102,106,2)

set basis cc-pvdz
set scf e_convergence=11
for R in Rvals:
    h2o.R = R
    for A in Avals:
        h2o.A = A
        energy('scf')
```

The declarations of `Rvals` and `Avals` are both completely standard Python syntax. Having named our molecule `h2o` we can then set the values of `R` and `A` within the loops. Note that we do not need the dollar sign to access the Python variable in this example; that is required only when using Python variables with the `set` keyword.

3.12 Tables of Results

The results of computations can be compactly tabulated with the `Table()` Psithon function. For example, in the following potential energy surface scan for water

```
molecule h2o {
  O
  H 1 R
  H 1 R 2 A
}

Rvals=[0.9,1.0,1.1]
Avals=range(100,102,2)

table=Table(rows=["R","A"], cols=["E(SCF)","E(SCS)","E(DFMP2)"])

set basis cc-pvdz

for R in Rvals:
    h2o.R = R
    for A in Avals:
        h2o.A = A
        energy('dfmp2')
        escf = get_variable('SCF TOTAL ENERGY')
        edfmp2 = get_variable('DF-MP2 TOTAL ENERGY')
        escsmp2 = get_variable('SCS-DF-MP2 TOTAL ENERGY')
        table[R][A] = [escf, escsmp2, edfmp2]

print table
relative=table.copy()
relative.absolute_to_relative()
print relative
```

we first define a table (on line 10) with two row indices and three column indices. As the potential energy scan is performed, the results are stored (line 22) and the final table is printed to the output file (line 24). The table is converted from absolute energies to relative energies (in kcal mol^{-1}) on line 26, before being printed again. The relative energies are reported with respect to the lowest value in each column. More examples of how to control the formatting of the tables can be found in the sample input files provided; see Appendix E for a complete listing.

3.13 Python Wrappers

The Python foundations of the `PSI4` driver and Psithon syntax permit many commonly performed post-processing procedures to be integrated into the `PSI4` suite. Among these are automated computations of interaction energies through `cp()`, of a model chemistry applied to a database of systems through `db()`, and of several model chemistries together approximating greater accuracy through `cbs()`. These are discussed separately in

Appendix G (also exists in html form with better formatting). Note that the options documented for Python functions are placed as arguments in the command that calls the function (formatted in this manual as **option =** `value`), not in the `set globals` block or with any other `set` command.

4 Basis Sets

Basis sets in `PSI4` are Gaussian functions (not Slater-type functions or plane waves), all-electron [no effective core potentials (ECPs)], and of Gaussian94 format (for ease of export from EMSL). Both spherical harmonic (5D/7F) and Cartesian (6D/10F) Gaussian functions are supported, but their mixtures are not, neither within a basis set (*e.g.*, 6D/7F) nor within a calculation (*e.g.*, cartesian for the orbital basis and spherical for the fitting basis). For built-in basis sets, the correct `SPHERICAL`/`CARTESIAN` value is set internally from the orbital basis.

4.1 Built-in Basis Sets

Tables 4 (Pople), 6–7 (Dunning), and 11 (other) summarize the orbital basis sets available in `PSI4`. These tables are arranged so that columns indicate degree of augmentation by diffuse functions (generally necessary for anions, excited states, and noncovalent interactions) and DTQ56 indicate the X = ζ levels available. Several intermediate levels of diffuse space between the customary non-augmented and augmented versions have been supplied for each basis set, including heavy-augmented and Truhlar's[3] calendar truncations described in Table 5. Fitting bases JKFIT (Table 8), RI (Table 9), and DUAL (Table 10) are available for methods incorporating density-fitting or dual-basis approximations. JKFIT sets are appropriate for fitting $(oo|$-type products, such as encountered in SCF theory and the electrostatics/exchange terms of SAPT. RI sets are appropriate for fitting $(ov|$-type products, such as encountered in MP2 and most SAPT terms. Citations for basis sets can be found in their definition files at `psi4/lib/basis` in the source. For basis set availability by element and the default value for keyword `PUREAM`, consult Appendix F.

4.2 User-Defined Basis Sets

There are three routes by which a basis set in G94 format can be introduced to `PSI4`'s notice.

- Install new basis set file into `PSI4` basis library.

  Copy the basis set definitions for all elements into a blank file. Exclamation points denote comments. As the first line of the file, add the word `spherical` or `cartesian` to indicate whether the basis set will run in (5D/7F) or (6D/10F). Name the file with

Table 4: Summary of Pople-style orbital basis sets available in PSI4.

no diffuse		heavy-augmented		augmented	
Basis Set	[Alias]	Basis Set	[Alias]	Basis Set	[Alias]
STO-3G					
3-21G					
6-31G		6-31+G		6-31++G	
6-31G(d)	[6-31G*]	6-31+G(d)	[6-31+G*]	6-31++G(d)	[6-31++G*]
6-31G(d_p)	[6-31G**]	6-31+G(d_p)	[6-31+G**]	6-31++G(d_p)	[6-31++G**]
6-311G		6-311+G		6-311++G	
6-311G(d)	[6-311G*]	6-311+G(d)	[6-311+G*]	6-311++G(d)	[6-311++G*]
6-311G(d_p)	[6-311G**]	6-311+G(d_p)	[6-311+G**]	6-311++G(d_p)	[6-311++G**]
6-311G(2d)		6-311+G(2d)		6-311++G(2d)	
6-311G(2d_p)		6-311+G(2d_p)		6-311++G(2d_p)	
6-311G(2d_2p)		6-311+G(2d_2p)		6-311++G(2d_2p)	
6-311G(2df)		6-311+G(2df)		6-311++G(2df)	
6-311G(2df_p)		6-311+G(2df_p)		6-311++G(2df_p)	
6-311G(2df_2p)		6-311+G(2df_2p)		6-311++G(2df_2p)	
6-311G(2df_2pd)		6-311+G(2df_2pd)		6-311++G(2df_2pd)	
6-311G(3df)		6-311+G(3df)		6-311++G(3df)	
6-311G(3df_p)		6-311+G(3df_p)		6-311++G(3df_p)	
6-311G(3df_2p)		6-311+G(3df_2p)		6-311++G(3df_2p)	
6-311G(3df_2pd)		6-311+G(3df_2pd)		6-311++G(3df_2pd)	
6-311G(3df_3pd)		6-311+G(3df_3pd)		6-311++G(3df_3pd)	

Note: Absolutely no commas are allowed in basis set specification. Use the underscore character instead.

Table 5: Levels of truncation for diffuse functions in standard basis sets.

Augmentation Level	Angular Momenta in the Diffuse Space[a]		Valid Basis Sets		
	Li–Kr Main Group	H & He	Dζ	Tζ	Qζ
aug-cc-pVXZ	s, p, $\cdots$, $\ell_{max}-2$, $\ell_{max}-1$, ℓ_{max}	s, p, $\cdots$, $\ell_{max}-1$	aDZ	aTZ	aQZ
[b]heavy-aug-cc-pVXZ	s, p, $\cdots$, $\ell_{max}-2$, $\ell_{max}-1$, ℓ_{max}		haDZ	haTZ	haQZ
jun-cc-pVXZ	s, p, $\cdots$, $\ell_{max}-2$, $\ell_{max}-1$		jaDZ	jaTZ	jaQZ
may-cc-pVXZ	s, p, $\cdots$, $\ell_{max}-2$			maTZ	maQZ
$\cdots$	s, p				aaQZ
cc-pVXZ			DZ	TZ	QZ

[a] Dζ has $\ell_{max}=2$ or d. Tζ has $\ell_{max}=3$ or f. Qζ has $\ell_{max}=4$ or g, *etc.*

[b] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

Table 6: Summary of Dunning orbital basis sets available in PSI4.

Basis Set	no diffuse	feb	mar	apr	may	jun	heavy-aug[a]	aug	d-aug
cc-pVXZ	DTQ56	——6	——56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	DTQ56
cc-pV(X+d)Z	DTQ56	——6	——56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	DTQ56
cc-pCVXZ	DTQ56	——6	——56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	DTQ56
cc-pCV(X+d)Z	DTQ56	——6	——56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	DTQ56
cc-pwCVXZ	DTQ5		——5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	DTQ5
cc-pwCV(X+d)Z	DTQ5		——5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	DTQ5

[a] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

Table 7: Summary of Dunning Douglas-Kroll basis sets available in PSI4.

Basis Set	no diffuse	feb	mar	apr	may	jun	heavy-aug[a]	aug	d-aug
cc-pVXZ-DK	DTQ5						DTQ5	DTQ5	
cc-pV(X+d)Z-DK									
cc-pCVXZ-DK	DTQ5						DTQ5	DTQ5	
cc-pCV(X+d)Z-DK									
cc-pwCVXZ-DK	–TQ5						–TQ5	–TQ5	
cc-pwCV(X+d)Z-DK									

[a] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

Table 8: Summary of Dunning JK-fitting basis sets available in PSI4.

Basis Set	no diffuse	feb	mar	apr	may	jun	heavy-aug[a]	aug	d-aug
cc-pVXZ-JKFIT[b]	DTQ5		——5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	
cc-pV(X+d)Z-JKFIT	DTQ5		——5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	
cc-pCVXZ-JKFIT[b]									
cc-pCV(X+d)Z-JKFIT									
cc-pwCVXZ-JKFIT[b]									
cc-pwCV(X+d)Z-JKFIT									

[a] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

[b] The JKFIT basis sets are designed in the cc-stub(X+d)Z framework that includes an additional set of *d*-fuctions for second-row *p*-block elements. Identical basis sets with the cc-stubXZ-JKFIT label are provided for convenience.

Table 9: Summary of Dunning MP2-fitting basis sets available in `PSI4`.

Basis Set	no diffuse	feb	mar	apr	may	jun	heavy-aug[a]	aug	d-aug
cc-pVXZ-RI	DTQ56	——6	––56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	
cc-pV(X+d)Z-RI	DTQ56	——6	––56	—Q56	–TQ56	DTQ56	DTQ56	DTQ56	
cc-pCVXZ-RI									
cc-pCV(X+d)Z-RI									
cc-pwCVXZ-RI	DTQ5		––5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	
cc-pwCV(X+d)Z-RI	DTQ5		––5	—Q5	–TQ5	DTQ5	DTQ5	DTQ5	

[a] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

Table 10: Summary of Dunning dual-basis helper basis sets available in `PSI4`.

Basis Set	no diffuse	feb	mar	apr	may	jun	heavy-aug[a]	aug	d-aug
cc-pVXZ-DUAL	–TQ						–TQ	DTQ	
cc-pV(X+d)Z-DUAL									
cc-pCVXZ-DUAL									
cc-pCV(X+d)Z-DUAL									
cc-pwCVXZ-DUAL									
cc-pwCV(X+d)Z-DUAL									

[a] The heavy-aug-cc-stub and jul-cc-stub basis sets are identical.

Table 11: Summary of other orbital basis sets available in `PSI4`.

Karlsruhe		Other
no diffuse	augmented	
def2-SV(P)		DZP
def2-SVP	def2-SVPD	TZ2P
def2-TZVP	def2-TZVPD	TZ2PF
def2-TZVPP	def2-TZVPPD	Sadlej-LPol-ds
def2-QZVP	def2-QZVPD	Sadlej-LPol-dl
def2-QZVPP	def2-QZVPPD	Sadlej-LPol-fs
		Sadlej-LPol-fl

the name of the basis set and a `.gbs` extension, after applying the following transformations.

All letters lowercase

Replace all * with s

Replace all + with p

Replace all ( ) , with _ (underscores replace parentheses and commas)

For example, basis 6-31++G** is stored in `6-31ppgss.gbs`, and cc-pV(D+d)Z is stored in `cc-pv_dpd_z.gbs`. Only one basis set may be specified per file. Copy the new basis set file into `$PSIDATADIR/basis`.

- Use new basis set file in arbitrary location.

  Copy the basis set definitions for all elements into a blank file. Exclamation points denote comments. As the first line of the file, add the basis set name in brackets. As the second line of the file, add the word `spherical` or `cartesian` to indicate whether the basis set will run in (5D/7F) or (6D/10F). The combination of [basis name], `PUREAM` value, and element basis set specifications forms a section, like the one shown below. Multiple basis sets can be specified in the same file by adding additional sections to the file. Specify the location of the new basis set file in a `PSI4` input file with the command `basis file path/to/basis.file`, where the path can be either relative or absolute.

```
[ sto-3g ]
cartesian
****
H     0
S   3   1.00
      3.42525091             0.15432897
      0.62391373             0.53532814
      0.16885540             0.44463454
****
O     0
S   3   1.00
    130.7093200              0.15432897
     23.8088610              0.53532814
      6.4436083              0.44463454
SP   3   1.00
      5.0331513             -0.09996723             0.15591627
      1.1695961              0.39951283             0.60768372
      0.3803890              0.70011547             0.39195739
****
```

- Include new basis set in input file.

Construct for a basis set a section like the one above that includes [basis name], `PUREAM` value, and element basis set specifications. Hash signs denote comments. Copy the section into a `PSI4` input file and surround it with the command `basis {...}`, as shown below. Multiple basis sets can be specified by adding additional sections within the surrounding brackets.

```
basis {
# basis set section like in snippet above goes here
# additional basis set sections follow
}
```

4.3 Specifying Basis Sets

The primary basis set is specified with option `BASIS`, which assigns, for instance, aug-cc-pVDZ to all atoms of the last-defined molecule through `set basis aug-cc-pVDZ` or `basis aug-cc-pVDZ` within a `set {...}` block. More details are provided in section 3.7.

5 Theoretical Methods Available in PSI4

Several electronic structure methods are available in the `PSI4` package, from Hartree–Fock molecular orbital theory to coupled-cluster theory to full configuration interaction. This section introduces the methods available and some of their most common input parameters. A complete list of standard keywords is provided in Appendix B, and keywords for expert users are listed in Appendix C.

5.1 General Options

Several options are relevant to many types of computations. Those are described briefly here.

`REFERENCE`	The reference wavefunction used in the computation **Possible Values:** RHF, ROHF, UHF, CUHF, RKS, UKS **Type:** string **Default:** RHF

The "reference wavefunction" specified by the `REFERENCE` keyword is the type of Hartree–Fock or Kohn–Sham wavefunction to be evaluated during the computation. The reference wavefunction may be the goal of the computation (for `energy('scf')`), or it may serve as a reference (zeroth-order solution) for various post-Hartree–Fock computations like CCSD(T) or MP2.

`PSI4` can use several types of reference wavefunctions. For closed-shell molecules, the appropriate choice is almost always restricted Hartree–Fock (RHF). For open-shell molecules, one can choose either restricted orbitals (restricted open-shell Hartree–Fock (ROHF)), or unrestricted orbitals (unrestricted Hartree–Fock (UHF)). We have also implemented the so-called constrained unrestricted Hartree–Fock (CUHF) method, which is an alternative approach to ROHF (and should give identical results) that may be easier to converge [8]. Not all post-Hartree–Fock modules in `PSI4` can use all possible reference types.

For Kohn–Sham density functional theory computations, the possible references are restricted Kohn–Sham (RKS, for closed-shell molecules) and unrestricted Kohn–Sham (UKS, for open-shell molecules). A restricted open-shell Kohn–Sham procedure is not recommended [9] and not implemented.

`DOCC`	An array containing the number of doubly-occupied orbitals per irrep (in Cotton order) **Type:** array **Default:** No Default
`SOCC`	An array containing the number of singly-occupied orbitals per irrep (in Cotton order) **Type:** array **Default:** No Default

DOCC and SOCC specify the electron configuration of interest by indicating how many doubly and singly occupied orbitals there are for each irreducible representation. These are arrays of integers, and the ordering is that given by Cotton. SOCC is only relevant for open-shell molecules. If DOCC and SOCC are not given, the program will attempt to guess the orbital occupations per irreducible representation automatically. For high-symmetry molecules, quantum chemistry programs like `PSI4` can frequently guess wrong, so having the ability to manually specify the correct configuration is useful. The automatic guess is less likely to be wrong if it is run in smaller basis sets (like STO-3G); doing so can be a practical way to get the right occupations, which can then be specified manually for larger-basis computations.

`FREEZE_CORE`	Specifies how many core orbitals to freeze in correlated computations. TRUE will default to freezing the standard default number of core orbitals. For heavier elements, there can be some ambiguity in how many core orbitals to freeze; in such cases, SMALL picks the most conservative standard setting (freezes fewer orbitals), and LARGE picks the least conservative standard setting (freezes more orbitals). More precise control over the number of frozen orbitals can be attained by using the keywords `NUM_FROZEN_DOCC` (gives the total number of orbitals to freeze, program picks the lowest-energy orbitals) or `FROZEN_DOCC` (gives the number of orbitals to freeze per irreducible representation) **Possible Values:** FALSE, TRUE, SMALL, LARGE

	Type: string **Default:** false
NUM_FROZEN_DOCC	The number of occupied orbitals to freeze in later correlated computations. FROZEN_UOCC trumps this option. **Type:** integer **Default:** 0
FROZEN_DOCC	The number of frozen doubly-occupied orbitals per irrep (these are not excited in a correlated wavefunction, nor can they be optimized in MCSCF). **Type:** array **Default:** No Default
NUM_FROZEN_UOCC	The number of virtual orbitals to freeze in later correlated computations. FROZEN_DOCC trumps this option. **Type:** integer **Default:** 0
FROZEN_UOCC	The number of frozen unoccupied orbitals per irrep (these are not populated in a correlated wavefunction, nor can they be optimized in MCSCF). **Type:** array **Default:** No Default
PRINT	The amount of information to print to the output file. 1 prints basic information, and higher levels print more information. A value of 5 will print very large amounts of debugging information. **Type:** integer **Default:** 1

5.2 Self-Consistent Field (SCF)

5.2.1 Introduction

Self-Consistent-Field (SCF) theory forms the cornerstone of *ab initio* quantum chemistry. Here SCF refers both to conventional Hartree–Fock (HF) molecular orbital theory and also to generalized Kohn–Sham Density Functional Theory (KS-DFT). PSI4 contains a mature code for several common HF references and a pilot (very shortly to be optimized) implementation of most flavors of KS-DFT.

An illustrative example of using the SCF module is as follows:

```
molecule {
0 3
O
O 1 1.21
}

set {
basis cc-pvdz
guess sad
reference uhf
scf_type pk
}

energy('scf')
```

This will run a UHF computation for triplet molecular oxygen (the ground state) using a PK algorithm for the Electron Repulsion Integrals (ERI) and starting from a Superposition of Atomic Densities (SAD) guess. After printing all manner of titles, geometries, sizings, and algorithm choices, the SCF finally reaches the iterations:

```
                        Total Energy        Delta E     Density RMS

   @UHF iter   0:  -149.76856421865352   -4.69109e+01   0.00000e+00
   @UHF iter   1:  -149.59793338958522    1.70631e-01   5.72371e-02
   @UHF iter   2:  -149.62408782458331   -2.61544e-02   8.04195e-03 DIIS
   @UHF iter   3:  -149.62679515182390   -2.70733e-03   2.51542e-03 DIIS
   @UHF iter   4:  -149.62726459105770   -4.69439e-04   1.06897e-03 DIIS
   @UHF iter   5:  -149.62730549814114   -4.09071e-05   2.70311e-04 DIIS
   @UHF iter   6:  -149.62730736371790   -1.86558e-06   5.94924e-05 DIIS
   @UHF iter   7:  -149.62730740227752   -3.85596e-08   9.93250e-06 DIIS
   @UHF iter   8:  -149.62730740325136   -9.73841e-10   1.88088e-06 DIIS
   @UHF iter   9:  -149.62730740326214   -1.07718e-11   1.80706e-07 DIIS
   @UHF iter  10:  -149.62730740326231   -1.70530e-13   2.19128e-08 DIIS
```

The algorithm takes 10 true iterations to converge the energy and density to the default of 1.0×10^{-8}, plus the trivial iteration due to the SAD guess. The energy on the zero-th iteration is not variational due to the improper idempotence properties of the SAD guess, but the first true iteration is within 2.0×10^{-4} relative error of the final answer, highlighting the efficiency of the SAD guess. The energy and density then converge smoothly, assisted by Pulay's Direct Inversion of the Iterative Subspace (DIIS), which is activated by default. DIIS from a high-quality guess is usually sufficient to converge the nonlinear SCF equations, however enhanced control of DIIS parameters and additional convergence algorithms are available and detailed below.

After the iterations are completed, a number of one-electron properties are printed, and some bookkeeping is performed to set up possible correlated computations. Additional one-electron properties are available by increasing the `PRINT` option. Also printed are the occupied and virtual orbital energies, which are useful in elucidating the stability and reactivity of the system. Finally, for UHF/UKS references, the S^2 diagnostic is printed, as the UHF/UKS ansatz does not constrain the wavefunction to be an eigenfunction of the $\hat{S}^2$ operator. In this case, the output shows that the current UHF wavefunction actually has an expectation value of $\langle S^2 \rangle = 2.03319$, which does not equal the expected eigenvalue of 2. If this metric rises significantly above a few hundredths, it may be necessary to switch to an ROHF reference, which is guaranteed to return a spin eigenfunction.

5.2.2 Background

In Hartree–Fock, the goal is to produce an optimized set of occupied molecular orbitals which describe the action of the electrons according to a one-particle Hamiltonian encapsulating electron-nuclear attraction, electronic kinetic energy, and mean-field electron-electron repulsion, all under the requirement of wavefunction antisymmetry. Molecular properties obtained by Hartree–Fock theory are generally at least qualitatively correct, although they can be quantitatively poor in many instances. If, as is usually the case, higher accuracy is desired, the occupied and virtual molecular orbitals are used as a starting point for the myriad of post-Hartree–Fock methods, all of which attempt to recover the exact correlations due to electron-electron repulsion.

DFT is an alternative approach which attempts to directly compute all observables as functionals of the electronic density. The method is tractable due to the two Hohenberg-Kohn Theorems. The first Theorem states that two electronic systems with the same ground state density are both bound by the same potential, to within a constant offset. This implies both that the ground-state density is sufficient to describe all ground state observables of the system and that the functional corresponding to electronic kinetic energy and electron-electron repulsion is universal. The second Theorem states that the exact energy functional is variational, obtaining its minimal value when the input density corresponds to the ground state density. This allows for a self-consistent computational procedure in which the density is varied until the energy functional is minimized. In practice, the exact energy functional is unknown (and would be as complex as Full Configuration Interaction if it were known), however many good approximations exist. A major problem with density-based DFT is that approximations to the kinetic energy functional are notoriously poor. This is usually overcome by the Kohn–Sham approach, which obtains most of the kinetic energy from a set of non-interacting orbitals constructed from the density. Within this approach, the kinetic energy, nuclear attraction energy, and mean-field Coulomb repulsion are treated in the same way as in Hartree–Fock, and the remaining exchange, correlation, and residual kinetic energy terms are treated via an approximate functional, typically built from some semi-local model of the exchange and Coulomb holes. In the generalized Kohn–Sham approach, the orbitals used to construct the non-interacting kinetic energy are also used to produce an orbital-dependent contribution to the exchange energy, either by adding a static fraction of the non-interacting exact exchange (the adiabatic connection or “hybrid” approach), or by partitioning the exchange hole into short range components be to treated by the local exchange functional, and long range components to be treated by the long-range noninteracting exact exchange (the range-separated approach).

5.2.3 Spin/Electronic Configuration Treatment

PSI4 solves the SCF equations in a basis of Gaussian functions using an iterative procedure. These equations attempt to determine the occupied molecular orbitals, and thereby ground state density, which minimize the Hartree–Fock or KS-DFT energy within the user-specified spin specialization and electronic configuration. Within Hartree–Fock, the allowed spin-specializations are Restricted-Hartree–Fock (RHF), Unrestricted-Hartree–Fock (UHF), Restricted-Open-Shell-Hartree–Fock (ROHF), and Constrained-Unrestricted-Hartree–Fock (CUHF). Within KS-DFT, the allowed spin-specializations are Restricted-Kohn–Sham (RKS) and Unrestricted-Kohn–Sham (UKS). Restricted-Open-Shell-Kohn–Sham (ROKS) and Constrained-Unrestricted-Kohn–Sham (CUKS) are not implemented, as they always predict a ground state with positive definite spin density, which is not physical. Spin specialization is specified by the REFERENCE option, which defaults to RHF. Electronic configuration is specified by the charge/multiplicity field in the molecular geometry input. If spatial symmetry is used, the assignment of occupied orbitals to irreps may be specified by the DOCC and SOCC integer arrays. If these arrays are not specified by the user, the program will assign occupation to irreps according to the lowest orbital energies at each SCF iteration (Aufbau ordering).

5.2.4 Orthogonalization

One of the first steps in the SCF procedure is the determination of an orthogonal basis (known as the OSO basis) from the atomic orbital basis (known as the AO basis). The Molecular Orbital basis (MO basis) is then built as a particular unitary transformation of the OSO basis. In PSI4, the determination of the OSO basis is accomplished via either symmetric or canonical orthogonalization. Symmetric orthogonalization uses the symmetric inverse square root of the overlap matrix for the orthogonalization matrix. Use of symmetric orthogonalization always yields the same number of OSO functions (and thereby MOs) as AO functions. However, this may lead to numerical problems if the overlap matrix has small eigenvalues, which may occur for large systems or for systems where diffuse basis sets are used. This problem may be avoided by using canonical orthogonalization, in which an asymmetric inverse square root of the overlap matrix is formed, with numerical stability enhanced by the elimination of eigenvectors corresponding to very small eigenvalues. As a few combinations of AO basis functions may be discarded, the number of canonical-orthogonalized OSOs and MOs may be slightly smaller than the number of AOs. In PSI4, symmetric orthogonalization is used by default, unless the smallest overlap eigenvalue falls below the user-supplied double option S_TOLERANCE, which defaults to 1.0×10^{-7}. If the smallest eigenvalue is below this cutoff, canonical orthogonalization is forced, and all eigenvectors corresponding to eigenvalues below the cutoff are eliminated. Use of canonical orthogonalization can be forced by setting the S_ORTHOGONALIZATION option to CANONICAL. Note that in practice, the MOs and OSOs are built separately within each irrep from the symmetry-adapted combinations of AOs known as Unique Symmetry Orbitals (USOs). For canonical orthogonalization, this implies that the number of MOs and OSOs per irrep may be slightly smaller than the number of USOs per irrep.

A contrived example demonstrating OSOs/MOs vs. AOs with symmetry is shown below:

```
Input:

molecule h2o {
0 1
O
H 1 1.0
H 1 1.0 2 104.5
symmetry c2 # Two irreps is easier to comprehend
}

set {
s_tolerance 0.0001      # Set an unreasonably tight
                        # tolerance to force canonical
basis aug-cc-pv5z       # This diffuse basis will have
                        # small-ish eigenvalues for even H2O
}

energy('scf')

Output:

  ... Initialization ...

  ==> Pre-Iterations <==

  Minimum eigenvalue in the overlap matrix is 1.6888059293E-05.
  Using Canonical Orthogonalization with cutoff of 1.0000000000E-04.
  Overall, 3 of 287 possible MOs eliminated.

  ... Initial Orbital Guess Information ...

   -------------------------------------------------------
    Irrep   Nso     Nmo     Nalpha   Nbeta   Ndocc  Nsocc
   -------------------------------------------------------
     A        145     144       3        3       3       0
     B        142     140       2        2       2       0
   -------------------------------------------------------
    Total     287     284       5        5       5       0
   -------------------------------------------------------
```

In this example, there are 287 AO basis functions after spherical harmonics are applied. These are used to produce 287 symmetry adapted USOs, 145 of which are assigned to irrep A, and 142 of which are assigned to irrep B. Within irrep A, 144 OSOs fall above the eigenvalue cutoff, and within irrep B 140 OSOs fall above the eigenvalue cutoff. In total, 284 molecular orbitals are chosen from 287 AOs/USOs. The table also shows the initial assignment of electrons to irreps.

5.2.5 Initial Guess/Convergence Stabilization

In each step of the SCF procedure, a new Fock or Kohn–Sham potential is built according to the previous density, following which the potential is diagonalized to produce new molecular orbitals, from which a new density is computed. This procedure is continued until either convergence is reached or a preset maximum number of iterations is exceeded. Convergence is determined by both change in energy and root-mean-square change in density matrix values, which must be below the user-specified `E_CONVERGENCE` and `D_CONVERGENCE`, respectively. The maximum number of iterations is specified by the `MAXITER` option. It should be noted that SCF is a chaotic process, and, as such, often requires careful selection of initial orbitals and damping during iterations to ensure convergence. This is particularly likely for large systems, metallic systems, multireference systems, open-shell systems, anions, and systems with diffuse basis sets.

For initial orbital selection, several options are available. The default guess is a core Hamiltonian which ignores even mean-field electronic repulsion. A much more accurate guess is the Superposition-of-Atomic-Densities (SAD) guess, which builds the initial molecular density as a sum of atomic UHF densities. As discussed above, the "zero-th" SCF iteration involving the SAD density need not be variational, and the resultant energy is usually garbage. However, the density obtained in the zero-th iteration is usually excellent, and the resultant first-iteration energy is generally less than a fraction of a percent off of the converged value. The SAD guess also produces fractionally occupied closed-shell orbitals, as required by the density-fitted exchange routines (discussed below). For open-shell systems, the Generalized Wolfsberg-Helmholtz (GWH) guess is available, and may be superior to a core guess. `SAD` and `GWH` may be invoked by the `GUESS` option. A more involved alternative to these approaches is to converge the SCF in a small basis and then project the resultant occupied molecular orbitals up into a larger basis. This can be done automatically by placing a **cast_up** = '*small_basis_name* modifier in the `energy()` procedure call. We recommend the 3-21G basis for the small basis due to its efficient mix of flexibility and compactness. An example of performing an RHF solution of water by SAD guessing in a 3-21G basis and then casting up to cc-pVTZ is shown below:

```
molecule h2o {
0 1
O
H 1 1.0
H 1 1.0 2 104.5
}

set {
basis cc-pvtz
guess sad
}

energy('scf', cast_up = '3-21G')
```

With regard to convergence stabilization, Pulay's Direct Inversion of the Iterative Subspace (DIIS) extrapolation and Gill's Maximum Overlap Method (MOM) are both implemented. DIIS uses previous iterates of the Fock Matrix together with an error criterion based on the orbital gradient to produce an informed estimate of the next Fock Matrix. DIIS is almost always necessary to converge the SCF procedure and is therefore turned on by default. In rare cases, the DIIS algorithm may need to be modified or turned off altogether, which may be accomplished via the options detailed below. MOM was developed to combat a particular class of convergence failure: occupation flipping. In some cases, midway though the SCF procedure, a partially converged orbital which should be occupied in the fully-optimized SCF solution has a slightly higher orbital eigenvalue than some other orbital which should be destined to be a virtual orbital. This results in the virtual orbital being spuriously occupied for one or more iterations. Sometimes this resolves itself without help, other times the occupation flips back and forth between two, four, or more orbitals. This is typically visible in the output as a non-converging SCF which eventually settles down to steady oscillation between two (or more) different total energies. This behavior can be ameliorated by choosing occupied orbitals by "shape" instead of by orbital eigenvalue, i.e., by choosing the set of new orbitals which looks most like some previously known "good" set. The "good" set is typically the occupied orbitals from an one of the oscillating iterations with the lowest total energy. For an oscillating system where the lowest total energy occurs on iterations $N, N+2, \ldots$, invoking `MOM_START` N can often rescue the convergence of the SCF. MOM can be used in concert with DIIS, though care should be taken to not turn MOM on until the oscillatory behavior begins. In some cases, a static mixing of Fock Matrices from adjacent iterations can quench oscillations. This mixing, known as "damping" can be activated by setting the `DAMPING_PERCENTAGE` keyword to a nonzero percent.

5.2.6 SCF ERI Algorithms

The key difficulty in the SCF procedure is treatment of the four-index ERI contributions to the Fock Matrix. A number of algorithms are available in `PSI4` for these terms. The algorithm is selected by the `SCF_TYPE` keyword, which defaults to `PK`.

`PK`: An in-core, presorted algorithm using exact ERIs. Quite fast for a zero-error algorithm if enough memory is available. Integrals are generated only once, and symmetry is utilized to reduce number of integrals.

OUT_OF_CORE: An out-of-core, unsorted algorithm using exact ERIs. Overcomes the memory bottleneck of the current PK algorithm. Integrals are generated only once, and symmetry is utilized to reduce number of integrals.

DIRECT: An in-core repeated integral evaluation algorithm using exact ERIs. Symmetry is used to reduce the number of integrals, and no disk is used. However, integral regeneration is quite costly, implying that this algorithm should be used only if there is not enough disk space for the OUT_OF_CORE algorithm.

DF: A density-fitted algorithm designed for computations with thousands of basis functions. This algorithm is highly optimized, and is threaded with a mixture of parallel BLAS and OpenMP. Note that this algorithm should use the -JKFIT series of auxiliary bases, *not* the -RI or -MP2FIT bases. The default guess for auxiliary basis set should work for all Dunning bases, otherwise the DF_BASIS_SCF keyword can be used to manually specify the auxiliary basis. This algorithm is preferred unless either absolute accuracy is required [$\gtrsim$CCSD(T)] or a -JKFIT auxiliary basis is unavailable for the primary basis/atoms involved.

For some of these algorithms, Schwarz and/or density sieving can be used to identify negligible integral contributions in extended systems. To activate sieving, set the INTS_TOLERANCE keyword to your desired cutoff (1.0×10^{-12} is recommended for most applications).

5.2.7 KS-DFT

KS-DFT is coming soon. A pilot code is already implemented in PSI4, but the efficiency and correctness of this code is by no means guaranteed. We expect to have a production-level KS-DFT code for energies and gradients by mid-March 2012. Contact Rob Parrish at CCMST (robparrish@gmail.com) with feature requests for the DFT code.

5.2.8 Recomendations

The SCF code is already quite flexible and powerful, with new features being added weekly. We have tried as much as possible to keep the number of options (documented in Appendix B.30) to a minimum, and to allow all options to be used in the presence of all other options. Below are some rough words of advice about using the SCF code for practical calculations:

- For GUESS, the SAD guess is usually your friend, even for open-shell systems (at the very least, it gets the right number of electrons, unlike some other programs). For instance, we have found that a simple SAD guess is often as good as doing a full SCF in a 3-21G basis and then performing a cast-up, at a fraction of the cost. However, SAD and DOCC/SOCC arrays do not play very well together at the moment.

- For wall time, DF beats OUT_OF_CORE, and PK beats DIRECT. Use DF unless you need absolute accuracy or do not have a -JKFIT auxiliary set for your primary basis/atom type. Then use OUT_OF_CORE unless you run out of disk space.

- For very large systems, you may need to relax E_CONVERGENCE and D_CONVERGENCE, as the error is extensive in the size of the system. 1.0×10^{-7} is usually sufficient.
- Don't mess with the convergence options unless convergence is a problem. We have optimized the parameters for efficiency over a wide array of system types.

5.3 Density-Fitted Second-Order Møller–Plesset Perturbation Theory (DFMP2)

5.3.1 Introduction

Second-Order Møller–Plesset Perturbation Theory (MP2) occupies a unique role in quantum chemistry due to its small-prefactor $\mathcal{O}(N^5)$ treatment of dynamic electron correlation. This unusually cheap *ab initio* treatment of electron correlation may be made even more efficient by means of the Density-Fitting (DF) approximation (also known as Resolution-of-the-Identity or RI), wherein the quadratic ov products in the bra- and ket- of the $(ov|ov)$-type Electron Repulsion Integrals (ERIs) appearing in MP2 are cast onto a linear-scaling auxiliary basis by least-squares fitting. Substitution of the DF factorization into the MP2 equations results in a formal scaling and prefactor reduction of MP2, and further speed gains are possible due to heavy utilization of matrix-multiplication kernels and minimal storage requirements in a DF approach. The method has been found to be quite robust and accurate, and it should be preferred unless extreme accuracy is required or a fitting basis is not defined for the primary basis and atom type encountered. In particular, we have found excellent efficiency and tractability gains when using DFMP2 in concert with a DFSCF reference. An efficient, threaded, disk-based DFMP2 code is available in PSI4 for all single reference types available in the SCF module.

An example utilization of the code is:

```
molecule h2o {
0 1
O
H 1 1.0
H 1 1.0 2 104.5
}

set basis cc-pvdz
set scf_type df
set freeze_core True

energy('dfmp2')
```

The energy() call executes the predefined DFMP2 procedure, first calling the SCF module with a default RHF reference and DF algorithm for the two-electron integrals. When the orbitals are converged, the DFMP2 module is launched, which forms the density-fitted $(Q|ov)$ integrals and then builds the full $(ov|ov)$ tensor in blocks, evaluating the contributions to the MP2 energy as it goes. A RHF-MP2 wavefunction is selected automatically due to the RHF reference. In this example, we freeze the core, both for efficiency and because split-valence bases like cc-pVDZ do not contain core correlation functions. The result looks something like:

```
----------------------------------------------------------
 ===================> MP2 Energies <====================
----------------------------------------------------------
 Reference Energy          =    -76.0213974789664633 [H]
 Singles Energy            =     -0.0000000000000001 [H]
 Same-Spin Energy          =     -0.0512503261762665 [H]
 Opposite-Spin Energy      =     -0.1534098129352447 [H]
 Correlation Energy        =     -0.2046601391115113 [H]
 Total Energy              =    -76.2260576180779736 [H]
----------------------------------------------------------
 =================> SCS-MP2 Energies <==================
----------------------------------------------------------
 SCS Same-Spin Scale       =      0.3333333333333333 [-]
 SCS Opposite-Spin Scale   =      1.2000000000000000 [-]
 SCS Same-Spin Energy      =     -0.0170834420587555 [H]
 SCS Opposite-Spin Energy  =     -0.1840917755222936 [H]
 SCS Correlation Energy    =     -0.2011752175810492 [H]
 SCS Total Energy          =    -76.2225726965475161 [H]
----------------------------------------------------------
```

The theory, breakdown of results, and common keywords used in DFMP2 are presented below.

5.3.2 Theory

Møller–Plesset Theory (MPn) or Many-Body Perturbation Theory (MBPT) through second order has the spin-orbital formula:

$$E_{\text{total}}^{(2)} = E_{\text{Reference}} - \frac{f_{ia} f_{ia}}{\epsilon_a - \epsilon_i} - \frac{1}{4} \frac{\langle ij||ab\rangle^2}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \tag{1}$$

Here i and j are occupied spin orbitals, a and b are virtual spin orbitals, f_{ia} are the ov Fock Matrix elements, ϵ are the orbital eigenvalues, and $\langle ij||ab\rangle$ are the antisymmetrized physicist's ERIs. For converged RHF and UHF references, the singles correction,

$$E_{\text{MBPT}}^{(1)} = -\frac{f_{ia} f_{ia}}{\epsilon_a - \epsilon_i}, \tag{2}$$

is zero due to the Brillioun Condition, and the first contribution to the perturbation series is at the second order:

$$E^{(2)}_{\mathrm{MBPT}} = -\frac{1}{4}\frac{\langle ij|ab\rangle^2}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j}. \tag{3}$$

In the DFMP2 module, the first-order contribution, or "singles energy" is always evaluated. This term is a significant contributor to the total second-order energy if a ROHF reference is used. In this case, we have chosen to use the ROHF-MBPT(2) ansatz, in which the ROHF orbitals are semicanonicalized, the resultant nonzero Fock matrix elements f_{ia} are used to form the singles amplitudes, and then the second-order amplitudes are formed with the semicanonical spin orbitals via the same machinery as a UHF-MP2. Note that the singles energy should be very close to zero for RHF and UHF references; if it is not, there is a good chance your orbitals are not well converged. Tighten the SCF `E_CONVERGENCE` and/or `D_CONVERGENCE` keywords and try again.

To increase the efficiency of MP2 energy evaluation, spin integration and simplification is carried out. This also allows for the identification of Same-Spin (SS) and Opposite-Spin (OS) terms for use in Grimme's Spin-Component Scaled (SCS) MP2. For RHF-MP2 (also labeled as RMP2), the spin-free equations are (note that the integrals are now chemist's integrals over spatial orbitals)

$$E^{(2)}_{\mathrm{MBPT,OS}} = -\frac{(ia|jb)(ia|jb)}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \tag{4}$$

and

$$E^{(2)}_{\mathrm{MBPT,SS}} = -\frac{[(ia|jb) - (ib|ja)](ia|jb)}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j}. \tag{5}$$

For UHF-MP2 (also labeled as UMP2) and the second-order contribution to ROHF-MBPT(2) using semicanonical orbitals, the spin-free equations are

$$E^{(2)}_{\mathrm{MBPT,OS}} = -\frac{(ia^{\alpha}|jb^{\beta})(ia^{\alpha}|jb^{\beta})}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \tag{6}$$

and

$$\begin{aligned} E^{(2)}_{\mathrm{MBPT,SS}} = &-\frac{1}{2}\frac{[(ia^{\alpha}|jb^{\alpha}) - (ib^{\alpha}|ja^{\alpha})](ia^{\alpha}|jb^{\alpha})}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \\ &-\frac{1}{2}\frac{[(ia^{\beta}|jb^{\beta}) - (ib^{\beta}|ja^{\beta})](ia^{\beta}|jb^{\beta})}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j}. \end{aligned} \tag{7}$$

Note that the UHF-MP2 equations use three classes of integrals, while the RHF-MP2 equations use only one class. Because of this, a UHF-MP2 or ROHF-MBPT(2) energy should take roughly three times as long as an RHF-MP2 energy.

5.3.3 Recommendations

All-in-all, DFMP2 should be a simple module to use, with few keywords (fully documented in the Appendix B.17). Some basic recommendations are included below:

- DFMP2 should be run with the *ov*-type RI or MP2FIT auxiliary basis sets, *not* the -JKFIT basis sets. The automatic basis selector should work fine for all of the Dunning bases (provided the auxiliary basis exists for the atom in question). If it does not, use the `DF_BASIS_MP2` keyword to manually specify the basis.
- DFMP2 likes memory. At a minimum, $2Q^2$ doubles are required, where Q is the size of the auxiliary basis set. However, there is one disk transpose of the $(Q|ov)$ tensor in the RHF-MP2 algorithm [two for UHF-MP2 and ROHF-MBPT(2)], so more memory will reduce seek times. If you notice DFMP2 using more memory than allowed, it is possible that the threaded three-index ERI computers are using too much overhead memory. Set the `DF_INTS_NUM_THREADS` to a smaller number to prevent this in this section (does not affect threaded efficiency in the rest of the code).
- DFMP2 likes disk. At a minimum, $2Qov$ doubles are required for RHF-MP2, and $4Qov$ doubles are required for UHF-MP2.
- DFMP2 likes threads. Some of the formation of the $(Q|ov)$ tensor relies on threaded BLAS (such as MKL) for efficiency. The main $\mathcal{O}(N^5)$ step is done via small/medium-sized DGEMMs inside of OpenMP, so make sure to set the `OMP_NESTED` environment variable to `FALSE` to prevent thread thrash (or just as well, do not define `OMP_NESTED` at all).
- Freezing core is good for both efficiency and correctness purposes. Freezing virtuals is not recommended.The DFMP2 module will remind you how many frozen/active orbitals it is using in a section just below the title.
- ROHF-MBPT(2) may be preferred to UHF-MP2, as the latter can suffer from severe spin contamination in some cases.
- MP2 is not suitable for systems with multireference character. The orbital energies will come together and an explosion will occur.

5.4 Configuration Interaction

Configuration interaction (CI) is one of the most general ways to improve upon Hartree–Fock theory by adding a description of the correlated motions of electrons. Simply put, a CI wavefunction is a linear combination of Slater determinants (or spin-adapted configuration state functions), with the linear coefficients being determined variationally via diagonalization of the Hamiltonian in the given subspace of determinants. For a "single-reference" CI based on reference function $|\Phi_0\rangle$, we can write the CI expansion as follows:

$$|\Psi\rangle = c_0|\Phi_0\rangle + \sum_i^{\text{occ}}\sum_a^{\text{vir}} c_i^a|\Phi_i^a\rangle + \sum_{i<j}^{\text{occ}}\sum_{a<b}^{\text{vir}} c_{ij}^{ab}|\Phi_{ij}^{ab}\rangle + \sum_{i<j<k}^{\text{occ}}\sum_{a<b<c}^{\text{vir}} c_{ijk}^{abc}|\Phi_{ijk}^{abc}\rangle + \cdots \quad (8)$$

The simplest standard CI method that improves upon Hartree–Fock is a CI that adds all singly $|\Phi_i^a\rangle$ and doubly $|\Phi_{ij}^{ab}\rangle$ substituted determinants (CISD) to the reference determinant

$|\Phi_0\rangle$. The CISD wavefunction has fallen out of favor because truncated CI wavefunctions are not size-extensive, meaning that their quality degrades for larger molecules. MP2 is a less expensive alternative giving results similar to those of CISD for small molecules, but the quality of MP2 does not degrade for larger molecules. Coupled-cluster singles and doubles (CCSD) is another size-extensive alternative; it is only slightly more costly computationally than CISD, but it typically provides significantly more accurate results.

The CI code in `PSI4` is described in detail in Reference [10]. For the reasons stated above, the CI code in `PSI4` is not optimized for CISD computations. Instead, emphasis has been placed on developing a very efficient program to handle more general CI wavefunctions which may be helpful in more challenging cases such as highly strained molecules or bond breaking reactions. `detci` is a fast, determinant-based CI module based upon the string formalism of Handy [11]. It can solve for restricted active space configuration interaction (RAS CI) wavefunctions as described by Olsen, Roos, Jorgensen, and Aa. Jensen [12]. Excitation-class selected multi-reference CI wavefunctions, such as second-order CI, can be formulated as RAS CI's. A RAS CI selects determinants for the model space as those which have no more than n holes in the lowest set of orbitals (called RAS I) and no more than m electrons in the highest set of orbitals (called RAS III). An intermediate set of orbitals, if present (RAS II), has no restrictions placed upon it. All determinants satisfying these rules are included in the CI.

`detci` is also very efficient at computing full configuration interaction wavefunctions, and it is used in this capacity in the complete-active-space self-consistent-field (CASSCF) code. Use of `detci` for CASSCF wavefunctions is described in another section of this manual.

As just mentioned, the `detci` module is designed for challenging chemical systems for which simple CISD is not suitable. Because CI wavefunctions which go beyond CISD (such as RAS CI) are fairly complex, typically the `detci` program will be used in cases where the tradeoffs between computational expense and completeness of the model space are nontrivial. Hence, the user is advised to develop a good working knowledge of multi-reference and RAS CI methods before attempting to use the program for a production-level project. This user's manual will provide only an elementary introduction to the most important keywords. Additional information is available in the complete list of keywords for `detci` provided in Appendices B.15 and C.12.

The division of the molecular orbitals into various subspaces such as RAS spaces, or frozen vs. active orbitals, etc., needs to be clear not only to `detci` , but also at least to the transformation program (and in the case of MCSCF, to other programs as well). Thus, orbital subspace keywords such as `RAS1`, `RAS2`, `RAS3`, `FROZEN_DOCC`, `FROZEN_UOCC`, `ACTIVE`, etc., should be set in the global section of input so they may also be read by other modules.

For single-reference CI computations, the easiest way to invoke a CI computation with `detci` is simply to call `energy()`, `optimize()`, etc., with the common name for that CI wavefunction, like `energy('cisd')` for a CISD single-point energy. The Python driver recognizes `cisd`, `cisdt`, and `cisdtq`. Higher order single-refernce CI wavefunctions, like those including singles through 6-fold excitations, can be invoked using numbers, like `ci6`. A full CI can be specifed by `fci`. More complicated CI computations, like RASCI, can be performed by setting the appropriate keywords and calling the module generically like `energy('detci')`. The latter approach will also work for any of the previously-mentioned CI wavefunctions for which the driver has built-in shortcuts, so long as the relevant options (especially `EX_LEVEL`) are set appropriately. Some examples of single-refence CI, RASCI, and full CI computations are provided in `psi4/samples/`.

5.4.1 Basic Keywords

`REFERENCE`	Reference wavefunction **Possible Values:** RHF, ROHF **Type:** string **Default:** RHF
`R_CONVERGENCE`	Convergence is achieved when the RMS of the error in the CI vector is less than this value. The default is 10^{-4} for energies and 10^{-7} for gradients. **Type:** double **Default:** 10^{-4}
`EX_LEVEL`	The CI excitation level **Type:** integer **Default:** 2
`FCI`	Do a full CI (FCI)? **Type:** boolean **Default:** false
`FROZEN_DOCC`	The number of frozen (doubly) occupied orbitals per irrep **Type:** array **Default:** The zero vector
`FROZEN_UOCC`	The number of frozen unoccupied orbitals per irrep **Type:** array **Default:** The zero vector
`MAXITER`	Maximum number of iterations to diagonalize the Hamiltonian. **Type:** integer **Default:** 12
`NUM_ROOTS`	number of CI roots to find **Type:** integer **Default:** 1
	Type: boolean **Default:** false
`ICORE`	Specifies how to handle buffering of CI vectors. A value of 0 makes the program perform I/O one RAS subblock at a time; 1 uses entire CI vectors at a time; and 2 uses one irrep block at a time. Values of 0 or 2 cause some inefficiency in the I/O (requiring multiple reads of the C vector when constructing H in the iterative subspace if `DIAG_METHOD` = <u>`SEM`</u>), but require less core memory.

	Type: integer **Default:** 1
`OPDM`	Compute one-particle density matrix if not otherwise required? **Type:** boolean **Default:** false
`TDM`	Compute the transition density? Note: only transition densities between roots of the same symmetry will be evaluated. `detci` does not compute states of different irreps within the same computation; to do this, lower the symmetry of the computation. **Type:** boolean **Default:** false
`DIPMOM`	Compute the dipole moment? **Type:** boolean **Default:** false
`MPN`	When this option is `TRUE`, DETCI will compute the MPn series out to kth order where k is determined by `MAX_NUM_VECS`. For open-shell systems (`REFERENCE`=`ROHF`), DETCI will compute the ZAPTn series. `GUESS_VECTOR` must be set to `UNIT`, `HD_OTF` must be set to `TRUE`, and `HD_AVG` must be set to `ORB_ENER`; these should happen by default for `MPN`=`TRUE`. **Type:** boolean **Default:** false

For larger computations, additional keywords may be required, as described in the `detci` sections of the Appendices, B.15 and C.12.

5.4.2 Arbitrary Order Perturbation Theory

The `detci` module is capable of computing energies for arbitrary order Møller–Plesset perturbation theory (MPn, for closed-shell systems with an RHF reference) and for Z-averaged perturbation theory (ZAPTn, open-shell systems with an ROHF reference). However, please note that these computations are essentially doing high-order CI (up to full CI) computations to obtain these results, and hence they will only be possible for very small systems (generally a dozen electrons or less).

The simplest way to run high-order perturbation theory computations is to call, e.g., `energy('mp10')` to invoke an MP10 computation or `energy('zapt25')` to invoke a ZAPT25 computation. This will automatically set several additional user options to their appropriate values. The program uses the Wigner $(2n + 1)$ rule to obtain higher-order energies from lower-order wavefunctions.

For the interested reader, the additional user options that are automatically set up by the calls above are as follows. A call like `energy('mp10')` sets `MPN` = `TRUE`. The program uses the Wigner $(2n + 1)$ rule by default (`MPN_WIGNER` = `TRUE`) and figures out what order of wavefunction is necessary to reach the desired order in the energy. The program then sets `MAX_NUM_VECS` to the required order in the wavefunction. By default, the requested nth order energy is saved as the current energy to the process environment. ZAPTN works essentially the same way for an ROHF reference.

5.4.3 Arbitrary Order Coupled-Cluster Theory

This feature is not yet released in the Beta1 version of the code.

The `detci` module is also capable of computing arbitrary-order coupled-cluster energies, using an approach similar to that of Hirata and Bartlett [13], or of Olsen [14]. Notably, the approach in `detci` also allows arbitrary-order *active space* coupled-cluster procedures. The general algorithm for doing this in `detci` is inefficient compared to optimized lower-order coupled-cluster codes and should not be used for CCSD, where the `ccenergy` module is much more efficient. For higher-order CC (like CCSDT and beyond), the code is also not as efficient as the MRCC code by Kállay, to which `PSI4` can interface (see Section 5.8); however, it may allow certain truncations of the model space that might not be available presently in MRCC. For very small systems, the code can be useful for testing of, for example, CCSDTQ or its active-space CCSDtq analog [15].

To perform arbitrary-order coupled-cluster, set the `detci` option `CC` to `TRUE`, and set `CC_EX_LEVEL` (note: not `EX_LEVEL`) to the desired coupled-cluster excitation level, and invoke `energy('detci')`. Various other `detci` options have a similar option for coupled-cluster, usually named beginning with CC. The full list of options is given in Appendices B.15 and C.12.

5.5 Coupled Cluster Methods

The coupled cluster approach is one of the most accurate and reliable quantum chemical techniques for including the effects of electron correlation. Instead of the linear expansion of the wavefunction used by configuation interaction, coupled cluster uses an exponential expansion,

$$\begin{aligned} |\Psi\rangle &= e^{\hat{T}}|\Phi_0\rangle \qquad (9) \\ &= \left(1 + \hat{T} + \frac{1}{2}\hat{T}^2 + \frac{1}{3!}\hat{T}^3 + \cdots\right)|\Phi_0\rangle, \end{aligned}$$

where the cluster operator $\hat{T}$ is written as a sum of operators that generate singly-excited, doubly-excited, etc., determinants:

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \cdots + \hat{T}_N, \qquad (10)$$

with

$$\hat{T}_1|\Phi_0\rangle = \sum_i^{\text{occ}} \sum_a^{\text{vir}} t_i^a |\Phi_i^a\rangle, \qquad (11)$$

$$\hat{T}_2|\Phi_0\rangle = \sum_{i<j}^{\text{occ}} \sum_{a<b}^{\text{vir}} t_{ij}^{ab} |\Phi_{ij}^{ab}\rangle, \qquad (12)$$

etc. The popular coupled cluster singles and doubles (CCSD) model [16] truncates the expansion at $\hat{T} = \hat{T}_1 + \hat{T}_2$. This model has the same number of parameters as configuration interaction singles and doubles (CISD) but improves upon it by approximately accounting for higher-order terms using products of lower-order terms (e.g., the term $\hat{T}_2^2$ approximately accounts for quadruple excitations). The inclusion of such products makes coupled-cluster methods *size extensive*, meaning that the quality of the computation should not degrade for larger molecules. The computational cost for CCSD scales as $\mathcal{O}(o^2v^4)$, where o is the number of occupied orbitals and v is the number of virtual orbitals.

Improving upon CCSD, the CCSD(T) method [17] includes a perturbative estimate of the energy contributed by the $\hat{T}_3$ operator. The computational cost of this additional term scales as $\mathcal{O}(o^3v^4)$, making it rather expensive for molecules with more than a dozen heavy atoms or so. However, when this method is affordable, it provides very high quality results in most cases.

PSI4 is capable of computing energies and analytic gradients for a number of coupled cluster models. It can also compute linear response properties (such as static or frequency-dependent polarizability, or optical rotation angles) for some models. Excited states can also be computed by the CC2 and CC3 models, or by EOM-CCSD. Table 12 summarizes these capabilities. This section describes how to carry out coupled cluster calculations within PSI4. For higher-order coupled-cluster methods like CCSDT and CCSDTQ, PSI4 can interface to Kállay's MRCC code (see Section 5.8).

Table 12: Current coupled cluster capabilities of PSI4.

Reference	Method	Energy	Gradient	Exc. Energies	LR Props
RHF	CC2	Y	N	Y	Y
UHF	CC2	Y	N	Y	N
ROHF	CC2	Y	N	Y	N
RHF	CCSD	Y	Y	Y	Y
RHF	CCSD(T)	Y	N	–	–
ROHF	CCSD	Y	Y	Y	N
ROHF	CCSD(T)	Y	N	–	–
UHF	CCSD	Y	Y	Y	N
UHF	CCSD(T)	Y	Y	–	–
RHF	CC3	Y	N	Y	N
UHF	CC3	Y	N	Y	N
ROHF	CC3	Y	N	Y	N
Brueckner	CCD	Y	N	N	N
Brueckner	CCD(T)	Y	N	–	–

The following wavefunctions are currently recognized by PSI4 as arguments to functions like `energy()`: `ccsd`, `ccsd(t)`, `cc2`, `cc3`, `bccd` (CCD with Brueckner orbitals), `bccd(t)` (CCD(T) with Brueckner orbitals), `eom-ccsd`, `eom-cc2` (CC2 for excited states), `eom-cc3` (CC3 for excited states). Response properties can be obtained by calling the function `response()` (instead of, for example, `energy()`), e.g., `response('ccsd')`. There are many sample coupled cluster inputs provided in `psi4/samples`.

5.5.1 Basic Keywords

A complete list of keywords related to coupled-cluster computations is provided in Appendix B, with the majority of the relevant keywords appearing in Appendix B.4. For a standard ground-state CCSD or CCSD(T) computation, the following keywords are common:

`REFERENCE` Reference wavefunction type
Possible Values: RHF, ROHF, UHF
Type: string **Default:** RHF

`R_CONVERGENCE` Convergence criterion for wavefunction (change) in CC amplitude equations.
Type: double **Default:** 1e-7

`MAXITER` Maximum number of iterations to solve the CC equations
Type: integer **Default:** 50

`BRUECKNER_ORBS_R_CONVERGENCE`
Convergence criterion for Breuckner orbitals. The convergence is determined based on the largest T_1 amplitude.
Type: double **Default:** 1e-5

`RESTART` Do restart the coupled-cluster iterations from old t_1 and t_2 amplitudes? For geometry optimizations, Brueckner calculations, etc. the iterative solution of the CC amplitude equations may benefit considerably by reusing old vectors as initial guesses. Assuming that the MO phases remain the same between updates, the CC codes will, by default, re-use old vectors, unless the user sets `RESTART = false`.
Type: boolean **Default:** true

`CACHELEVEL` Cacheing level for libdpd governing the storage of amplitudes, integrals, and intermediates in the CC procedure. A value of 0 retains no quantities in cache, while a level of 6 attempts to store all quantities in cache. For particularly large calculations, a value of 0 may help with certain types of memory problems. The default is 2, which means that all four-index quantites with up to two virtual-orbital indices (e.g., $\langle ij|ab\rangle >$ integrals) may be held in the cache.

	Type: integer **Default:** 2
`CACHETYPE`	Selects the priority type for maintaining the automatic memory cache used by the libdpd codes. A value of LOW selects a "low priority" scheme in which the deletion of items from the cache is based on pre-programmed priorities. A value of LRU selects a "least recently used" scheme in which the oldest item in the cache will be the first one deleted. **Possible Values:** LOW, LRU **Type:** string **Default:** LOW
`NUM_AMPS_PRINT`	Number of important t_1 and t_2 amplitudes to print **Type:** integer **Default:** 10
`MP2_AMPS_PRINT`	Do print the MP2 amplitudes which are the starting guesses for RHF and UHF reference functions? **Type:** boolean **Default:** false

5.5.2 Larger Calculations

Here are a few recommendations for carrying out large-basis-set coupled cluster calculations with `PSI4`:

1. In most cases it is reasonable to set the `MEMORY` keyword to 90% of the available physical memory, at most. There is a small amount of overhead associated with the coupled cluster modules that is not accounted for by the internal CC memory handling routines. Thus, the user should *not* sepcify the entire physical memory of the system, or swapping is likely. However, for especially large calculations, it is better to set the `MEMORY` keyword to a value less than 16 GB.

2. Set the `CACHELEVEL` keyword to `0`. This will turn off cacheing, which, for very large calculations, can lead to heap fragmentation and memory faults, even when sufficient physical memory exists.

3. Set the `PRINT` keyword to `2`. This will help narrow where memory bottlenecks or other errors exist in the event of a crash.

5.5.3 Excited State Coupled Cluster Calculations

A complete list of keywords related to coupled cluster linear response is provided in Appendices B.5 and C.4. The most important keywords associated with EOM-CC calculations are:

STATES_PER_IRREP

The number of electronic states to computed, per irreducible representation. Irreps denote the final state symmetry, not the symmetry of the transtion.

Type: array **Default:** none

E_CONVERGENCE

Convergence criterion for excitation energy (change) in the Davidson algorithm for CC-EOM.

Type: double **Default:** 1e-8

SINGLES_PRINT

Do print information on the iterative solution to the single-excitation EOM-CC problem used as a guess to full EOM-CC?

Type: boolean **Default:** false

SCHMIDT_ADD_RESIDUAL_TOLERANCE

Minimum absolute value above which a guess vector to a root is added to the Davidson algorithm in the EOM-CC iterative procedure.

Type: double **Default:** 1e-3

EOM_GUESS

Specifies a set of single-excitation guess vectors for the EOM-CC procedure. If `EOM_GUESS` = `SINGLES`, the guess will be taken from the singles-singles block of the similarity-transformed Hamiltonian, Hbar. If `EOM_GUESS` = `DISK`, guess vectors from a previous computation will be read from disk. If `EOM_GUESS` = `INPUT`, guess vectors will be specified in user input. The latter method is not currently available.

Possible Values: SINGLES, DISK, INPUT

Type: string **Default:** SINGLES

5.5.4 Linear Response (CCLR) Calculations

Linear response computations are invoked like `response('ccsd')` or `response('cc2')`. A complete list of keywords related to coupled cluster linear response is provided in Appendix B.8.

The most important keywords associated with CC-LR calculations are as follows.

PROPERTY

The response property desired. Acceptable values are `POLARIZABILITY` (default) for dipole-polarizabilities, `ROTATION` for specific rotations, `ROA` for Raman Optical Activity, and `ALL` for all of the above.

Possible Values: POLARIZABILITY, ROTATION, ROA, ALL

Type: string **Default:** POLARIZABILITY

`OMEGA` Array that specifies the desired frequencies of the incident radiation field in CCLR calculations. If only one element is given, the units will be assumed to be atomic units. If more than one element is given, then the units must be specified as the final element of the array. Acceptable units are `HZ`, `NM`, `EV`, and `AU`.
Type: array **Default:** 0

`GAUGE` Specifies the choice of representation of the electric dipole operator. Acceptable values are `LENGTH` for the usual length-gauge representation, `VELOCITY` for the modified velocity-gauge representation in which the static-limit optical rotation tensor is subtracted from the frequency-dependent tensor, or `BOTH`. Note that, for optical rotation calculations, only the choices of `VELOCITY` or `BOTH` will yield origin-independent results.
Possible Values: LENGTH, VELOCITY, BOTH
Type: string **Default:** LENGTH

5.6 *Ab initio* Polarization Propagator

The `adc` code seeks the pole structure of the polarization propagator, which is equivalent to the correlated excitation energy, in terms of the second order algebraic-diagrammatic construction (ADC(2)) scheme. Originally, the ADC scheme was derived in purely the diagrammatic language by Schirmer [18] and later, a sophisticated algebraic scheme was developed [19] by Trofimov et al. In general by n-th order ADC theory, the excited state is treated at completely equivalent level to the Møller–Plesset perturbation expansion of the same order. Hence the ADC(2) can be described as MP2 theory for the excited state and the relation to such other response theories as CC2-LR, CIS(D) and CIS(D_n) has been addressed [20] by Hättig et al. In the ADC theory, the residue calculus of the propagator is translated into the eigenvalue problem with respect to the correlated response matrix, also known as the shifted-Hamiltonian. The σ-vectors (Hamiltonian-vector products) are constructed several times in the simultaneous expansion method (SEM) to solve the eigenvalue problem, and each σ-vector construction has a computational cost that scales as $\mathcal{O}(N^5)$. In addition, the tensorial form of the σ-vector resembles to that of the doubles correction in the CIS(D) energetic equation. As a consequence, the pre-factor in the polynomial scaling becomes far larger than that of the CIS(D) even though the quasi-degeneracy of the excited state is properly accounted for in the ADC(2) model.

In `PSI4` the quite efficient and flexible integral-transformation library named `libtrans` is newly equipped and utilized in the production level DCFT code. The `adc` code is also based on `libtrans`, and it is also based on `libdpd`, a library to utilize molecular symmetry in the tensorial manipulations in framework of the direct-product decomposition algorithm. By this feature, the Ritz space and intermediate tensors are blocked according to the irreducible representations of the point group, and the excited states that belong to different symmetry are sought separately.

In the output of `adc` the ADC(2) results may look as follows:

```
->  1 B1 state   :  0.2565095 (a.u.),  6.9799824 (eV)
    Non-iterative:  0.2565636 (a.u.),  6.9814532 (eV)
             Occ Vir        Coefficient
    ---------------------------------------------
              3   0        -0.9017047264
              3   2         0.3038332241
              3   1         0.2907567119
              3   5        -0.0790167706
              3   4        -0.0425829926

    Converged in   4 iteration.
    Squared norm of the S component:  0.9315336
    The S vector is rotated up to  8.102 (deg.)
```

in which the ADC(2) excitation energy is indicated with arrow symbol and the pseudo-perturbative value, which is calculated in very similar fashion to the CIS(D) energy, is also presented on the following line. In this implementation, the ADC(2) secular matrix is treated effectively by renormalization of the double excitation manifold into the single excitation manifold. So, the effective secular equation is solved for several times for the specific state due to the eigenvalue dependence of the effective response matrix. Only the S component of the transition amplitude is obtained explicitly and the squared norm of the S block and the rotation angle from the corresponding CIS vector are given below the element of the amplitude. The difference between the ADC(2) value and its non-iterative counterpart is mostly negligible if the mixture among the CIS excited states is small and the quasi-degeneracy in the excited state is tolerably weak. But if there is a significant discrepancy in these energies, or the rotation angle is visibly large, special care may have to be taken for the strong effects caused by the higher excited states.

5.6.1 Partial Renormalization Scheme

The `adc` code is capable of performing the partially-renormalized ADC(2) computation, termed PR-ADC(2). In the perturbative treatment of the singly-excited state, the doubly and triply excited configurations are accounted for as in the case of CIS(D). In the language of CIS(D), the former is regarded to introduce the orbital relaxation (OR) effect while the latter is argued to give rise to the differential correlation (DC) correction to the excited state. In the PR-ADC(2) scheme, the the DC term is corrected according to the ground state PR-MP2 correlation, in which the correlation between the electron pairs is accounted for in size-consistent and unitary-invariant fashion by modulating the MP1 amplitude. By utilizing the `PR` scheme, substantial resistance against quasi-degeneracy is readily granted as discussed in Ref. [21].

5.6.2 Using the ADC(2) code

A complete list of keywords related to ADC(2) computations is provided in Appendix B.2. Some sample inputs are provided in `psi4/samples`, in directories starting with the name `adc`. The most important keyword is `STATES_PER_IRREP`, which is an array giving the number of excited states desired for each irreducible representation.

5.6.3 Implementation

Some very essential points are emphasized for understanding of the nature and the limitations of the theory. The ADC(2) response matrix, denoted as $\mathbf{A}$, is expanded in the single (S) and double (D) excitation manifolds as

$$\begin{pmatrix} \mathbf{A}^{(2)}_{\mathbf{SS}} & \mathbf{A}^{(1)}_{\mathbf{SD}} \\ \mathbf{A}^{(1)}_{\mathbf{DS}} & \mathbf{A}^{(0)}_{\mathbf{DD}} \end{pmatrix} \begin{pmatrix} \mathbf{X_S} \\ \mathbf{X_D} \end{pmatrix} = \omega \begin{pmatrix} \mathbf{X_S} \\ \mathbf{X_D} \end{pmatrix}$$

where the superscript on each matrix block indicates the order of the fluctuation. Instead of solving the above equation explicitly, the large D manifold is treated effectively as

$$[\mathbf{A}^{(2)}_{\mathbf{SS}} + {\mathbf{A}^{(1)}_{\mathbf{SD}}}^{\dagger}(\omega - \mathbf{A}^{(0)}_{\mathbf{DD}})^{-1}\mathbf{A}^{(1)}_{\mathbf{DS}}]\mathbf{X_S} = \omega\mathbf{X_S}.$$

This form of the ADC(2) equation requires $7 - 10$ iterations for convergence on only one root. But thanks to Newton-Raphson acceleration,

$$\omega^{n+1} = \omega^{n} - \frac{\omega^{n} - \mathbf{X_S}(\omega^{n})^{\dagger}[\mathbf{A}^{(2)}_{\mathbf{SS}} + {\mathbf{A}^{(1)}_{\mathbf{SD}}}^{\dagger}(\omega^{n} - \mathbf{A}^{(0)}_{\mathbf{DD}})^{-1}\mathbf{A}^{(1)}_{\mathbf{DS}}]\mathbf{X_S}(\omega^{n})}{1 + \mathbf{X_S}(\omega^{n})^{\dagger}[{\mathbf{A}^{(1)}_{\mathbf{SD}}}^{\dagger}(\omega^{n} - \mathbf{A}^{(0)}_{\mathbf{DD}})^{-2}\mathbf{A}^{(1)}_{\mathbf{DS}}]\mathbf{X_S}(\omega^{n})}$$

the computational time reduces to shorter than half of the simple iterative procedure. Construction of the denominator of the second term in the above equation is less computationally expensive than contruction of one σ-vector with respect to the effective response matrix. The non-iterative excitation energy stated above is calculated as a diagonal element of the Davidson mini-Hamiltonian matrix in the SEM as,

$$\omega^{Non-Iterative} = {\mathbf{X_{CIS}}}^{\dagger}[\mathbf{A}^{(2)}_{\mathbf{SS}} + {\mathbf{A}^{(1)}_{\mathbf{SD}}}^{\dagger}(\omega^{CIS} - \mathbf{A}^{(0)}_{\mathbf{DD}})^{-1}\mathbf{A}^{(1)}_{\mathbf{DS}}]\mathbf{X_{CIS}}$$

where ω^{CIS} and $\mathbf{X_{CIS}}$ denote the CIS excitation energy and wave function, respectively. The explicit form of the σ-vector is provided in a note accompanying the source code, in the file `src/bin/adc/sigma.pdf`.

5.6.4 The Density-Fitted Version

The density-fitted ADC(2) is given as a plugin `plugin_dfadc`, which is still under development. At present, it requires quite large amounts of memory in order to store the DF vectors. Significantly more refinement in algorithm may be necessary before this version is ready for routine use on large molecular systems.

5.7 Symmetry-Adapted Perturbation Theory

Symmetry-adapted perturbation theory (SAPT) provides a means of directly computing the noncovalent interaction between two molecules, that is, the interaction energy is determined without computing the total energy of the monomers or dimer. In addition, SAPT provides a decomposition of the interaction energy into physically meaningful components: *i.e.* electrostatic, exchange, induction, and dispersion terms. In SAPT, the Hamiltonian of the dimer is partitioned into contributions from each monomer and the interaction.

$$H = F_A + W_A + F_B + W_B + V \tag{13}$$

Here, the Hamiltonian is written as a sum of the usual monomer Fock operators, F, the fluctuation potential of each monomer, W, and the interaction potential, V. The monomer Fock operators, $F_A + F_B$, are treated as the zeroth-order Hamiltonian and the interaction energy is evaluated through a perturbative expansion of V, W_A, and W_B. Through first-order in V, electrostatic and exchange interactions are included; induction and dispersion first appear at second-order in V. For a complete description of SAPT, the reader is refered to the excellent review by Jeziorski, Moszynski, and Szalewicz [1].

Several truncations of the SAPT expansion are available in the `sapt` module of `PSI4`. The simplest truncation of SAPT is denoted SAPT0.

$$E_{SAPT0} = E^{(10)}_{elst} + E^{(10)}_{exch} + E^{(20)}_{ind,resp} + E^{(20)}_{exch-ind,resp} + E^{(20)}_{disp} + E^{(20)}_{exch-disp}. \tag{14}$$

In this notation, $E^{(vw)}$ defines the order in V and in $W_A + W_B$; the subscript, *resp*, indicates that orbital relaxation effects are included.

$$E_{SAPT2} = E_{SAPT0} + E^{(12)}_{elst,resp} + E^{(11)}_{exch} + E^{(12)}_{exch} + {}^{t}E^{(22)}_{ind} + {}^{t}E^{(22)}_{exch-ind} \tag{15}$$

$$E_{SAPT2+} = E_{SAPT2} + E^{(21)}_{disp} + E^{(22)}_{disp} \tag{16}$$

$$E_{SAPT2+(3)} = E_{SAPT2+} + E^{(13)}_{elst,resp} + E^{(30)}_{disp} \tag{17}$$

$$E_{SAPT2+3} = E_{SAPT2+(3)} + E^{(30)}_{exch-disp} + E^{(30)}_{ind-disp} + E^{(30)}_{exch-ind-disp} \tag{18}$$

A thorough analysis of the performance of these truncations of SAPT can be found in a review by Hohenstein and Sherrill [2].

The `sapt` module relies entirely on the density-fitting approximation of the two-electron integrals. The factorization of the SAPT energy expressions, as implemented in `PSI4`, assumes the use of density-fitted two-electron integrals, therefore, the `sapt` module cannot be run with exact integrals. In practice, we have found that the density-fitting approximation introduces negligable errors into the SAPT energy and greatly improves efficiency.

5.7.1 A First Example

The following is the simplest possible input that will perform all available SAPT computations (normally, you would pick one of these methods).

```
molecule water_dimer {
     0 1
     O  -1.551007  -0.114520   0.000000
     H  -1.934259   0.762503   0.000000
     H  -0.599677   0.040712   0.000000
     --
     0 1
     O   1.350625   0.111469   0.000000
     H   1.680398  -0.373741  -0.758561
     H   1.680398  -0.373741   0.758561

     units angstrom
     no_reorient
     symmetry c1
}

set globals {
    basis         aug-cc-pvdz
}

energy('sapt0')
energy('sapt2')
energy('sapt2+')
energy('sapt2+(3)')
energy('sapt2+3')
```

The `sapt` module uses the standard `PSI4` partitioning of the dimer into monomers. Additionally, the `no_reorient` flag must be included and the use of spatial symmetry disabled by setting the `symmetry` option to `c1`. A final note is that the `sapt` module is only capable of performing SAPT comuptations for interactions between closed-shell singlets.

The example input shown above would not be used in practice. To exploit the efficiency of the density-fitted SAPT implementation in `PSI4`, the SCF computations should also be performed with density-fitted (DF) integrals.

```
set globals {
    basis         aug-cc-pvdz
    df_basis_scf  aug-cc-pvdz-jkfit
    df_basis_sapt aug-cc-pvdz-ri
    guess         sad
    scf_type      df
}

set sapt {
    print         1
}
```

These options will perform the SAPT computation with DF-HF and a superposition-of-atomic-densities guess. This is the preferred method of running the `sapt` module.

5.7.2 SAPT0

Generally speaking, SAPT0 should be applied to large systems or large data sets. The performance of SAPT0 relies entirely on error cancellation, which seems to be optimal with a truncated aug-cc-pVDZ basis, namely, jun-cc-pVDZ (which we have referred to in previous work as aug-cc-pVDZ′). The `sapt` module has been used to perform SAPT0 computations with over 200 atoms and 2800 basis functions; this code should be scalable to 4000 basis functions. Publications resulting from the use of the SAPT0 code should cite the following publications:

E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.*, **132**, 184111 (2010).

E. G. Hohenstein, R. M. Parrish, C. D. Sherrill, J. M. Turney, and H. F. Schaefer, *J. Chem. Phys.*, **135**, 174107 (2011).

Basic SAPT0 Keywords

`BASIS`	The basis set used to describe the monomer molecular orbitals. **Type:** string	**Default:** none
`DF_BASIS_SAPT`	The fitting basis to use for all two-electron integrals in the SAPT computation. `PSI4` will attempt to pick a reasonable fitting basis if one is not provided. **Type:** string	**Default:** none
`DF_BASIS_ELST`	Optionally, a different fitting basis can be used for the $E^{(10)}_{elst}$ and $E^{(10)}_{exch}$ terms. This may be important if heavier elements are involved.	

	Type: string **Default:** none
`FREEZE_CORE`	Sets the number of core orbitals to freeze in the evaluation of the $E^{(20)}_{disp}$ and $E^{(20)}_{exch-disp}$ terms. It is recommended to freeze core in all SAPT computations. **Possible Values:** TRUE, FALSE, SMALL, LARGE **Type:** string **Default:** FALSE
`D_CONVERGENCE`	Convergence of the residual of the CPHF coefficients needed for the $E^{(20)}_{ind,resp}$ term. **Type:** double **Default:** 1.0×10^{-8}
`E_CONVERGENCE`	Convergence of the energy change in the $E^{(20)}_{ind,resp}$ term during the solution of the CPHF equations (in hartrees). **Type:** double **Default:** 1.0×10^{-10}
`MAXITER`	The maximum number of CPHF iterations. **Type:** integer **Default:** 50
`PRINT`	The print level for the `sapt` module. If `PRINT` is set to 0, only the header and final results are printed. If `PRINT` is set to 1, some intermediate quantities are also printed. For large SAPT computations, it is advisable to set `PRINT` to 1 so the progess of the computation can be tracked. **Type:** integer **Default:** 1

Advanced SAPT0 Keywords

`AIO_CPHF`	Do disk I/O asynchronously during the solution of the CPHF equations. This option may speed up the computation slightly, however use of this option will cause `PSI4` to spawn an additional thread. **Type:** boolean **Default:** FALSE
`AIO_DF_INTS`	Do disk I/O asynchronously during the formation of the DF integrals. This option may speed up the computation slightly, however use of this option will cause `PSI4` to spawn an additional thread. **Type:** boolean **Default:** FALSE
`NO_RESPONSE`	Don't solve the CPHF equations, evaluate $E^{(20)}_{ind}$ and $E^{(20)}_{exch-ind}$ instead of their response-including counterparts. Only turn on this option if you are not going to use the induction energy.

	Type: boolean **Default:** FALSE
`INTS_TOLERANCE`	All three-index DF integrals and those contributing to four-index integrals that fall below this Schwarz bound will be neglected. The default is very conservative, however, there isn't much to gain from loosening it. **Type:** double **Default:** 1.0×10^{-12}
`DENOMINATOR_DELTA`	The `sapt` module uses approximate energy denominators for most of the $E^{(20)}_{disp}$ and $E^{(20)}_{exch-disp}$ evaluation. This option controls the maximum allowable error norm in the energy denominator tensor. **Type:** double **Default:** 1.0×10^{-6}
`DENOMINATOR_ALGORITHM`	Should the energy denominators be approximated with Laplace transformations or a Cholesky decomposition? We have found Laplace transformations to be slightly more efficient. **Possible Values:** LAPLACE, CHOLESKY **Type:** string **Default:** LAPLACE
`SAPT_OS_SCALE`	The `sapt` module will print a decomposition of the $E^{(20)}_{disp}$ and $E^{(20)}_{exch-disp}$ terms into same-spin and oppposite-spin contributions, in analogy to the SCS-MP2 method of Stefan Grimme. This option controls the scaling of the oppposite-spin contributions. **Type:** double **Default:** 6.0/5.0
`SAPT_SS_SCALE`	This option controls the scaling of the same-spin contributions. **Type:** double **Default:** 1.0/3.0
`DEBUG`	The `DEBUG` flag will print lots of additional intermediate quantities that are not usually interesting. It will also do additional work (which is not optimized for large systems). Don't turn on the `DEBUG` flag. **Type:** integer **Default:** 0

5.7.3 Higher-order SAPT

For smaller systems (up to the size of a nucleic acid base pair), more accurate interaction energies can be obtained through higher-order SAPT computations. The `sapt` module can perform density-fitted evaluations of SAPT2, SAPT2+, SAPT2+(3), and SAPT2+3 energies. Publications resulting from the use of the higher-order SAPT code should cite the following:

E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.*, **133**, 014101 (2010).

A brief note on memory usage: the higher-order SAPT code assumes that certain quantities can be held in core. This code requires sufficient memory to hold $3o^2v^2 + v^2N_{aux}$ arrays in core. With this requirement computations on the adenine-thymine complex can be performed with an aug-cc-pVTZ basis in less than 64GB of memory.

Higher-order SAPT is treated separately from the higly optimized SAPT0 code, therefore, higher-order SAPT uses a separate set of keywords. The following keywords are relevant for higher-order SAPT.

Basic Keywords for Higher-order SAPT

`BASIS`	The basis set used to describe the monomer molecular orbitals. **Type:** string **Default:** none
`DF_BASIS_SAPT`	The fitting basis to use for all two-electron integrals in the SAPT computation. `PSI4` will attempt to pick a reasonable fitting basis if one is not provided. **Type:** string **Default:** none
`FREEZE_CORE`	Sets the number of core orbitals to freeze in the evaluation of all dispersion terms and intramonomer correlation amplitudes. It is highly recommended to freeze core in all higher-order SAPT computations. **Possible Values:** TRUE, FALSE, SMALL, LARGE **Type:** string **Default:** FALSE
`PRINT`	The print level for the `sapt` module. If `PRINT` is set to 0, only the header and final results are printed. If `PRINT` is set to 1, some intermediate quantities are also printed. For large SAPT computations, it is advisable to set `PRINT` to 1 so the progess of the computation can be tracked. **Type:** integer **Default:** 1

Advanced Keywords for Higher-order SAPT

`INTS_TOLERANCE`	All three-index DF integrals and those contributing to four-index integrals that fall below this Schwarz bound will be neglected. The default is very conservative, however, there is nothing to gain from loosening it in the case of higher-order SAPT. **Type:** double **Default:** 1.0×10^{-12}
`SAPT_MEM_CHECK`	This flag can be used to disable memory checking in the higher-order SAPT code; disabling it is ill advised.

	Type: boolean **Default:** TRUE
DEBUG	The DEBUG flag will print lots of additional intermediate quantities that are not usually interesting. **Type:** integer **Default:** 0

5.7.4 MP2 Natural Orbitals

One of the unique features of the sapt module is its ability to use MP2 natural orbitals (NOs) to speed up the evaluation of the triples contribution to disperison. By transforming to the MP2 NO basis, we can throw away virtual orbitals that are expected to contribute little to the dispersion energy. Speedups in excess of 50× are possible. In practice, this approximation is very good and should always be applied. Publications resulting from the use of MP2 NO-based approximations should cite the following:

E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.*, **133**, 104107 (2010).

Basic Keywords Controlling MP2 NO Approximations

NAT_ORBS	This flag activates MP2 NO approximations for the triples correction to dispersion. **Type:** boolean **Default:** FALSE
OCC_TOLERANCE	All virtual orbitals with smaller occupation numbers (eigenvalues of the MP2 one-particle density matrix) than this threshold will be discarded. See the above reference for a discussion of acceptable values for OCC_TOLERANCE. **Type:** double **Default:** 1.0×10^{-6}

Advanced Keywords Controlling MP2 NO Approximations

NAT_ORBS_T2	This flag activates MP2 NO approximations for the v^4 block of two-electron integrals in the evaluation of second-order T2 amplitudes. At present, this approximation has not been rigorously tested, however, initial results are promising for both accuracy and computational savings. **Type:** boolean **Default:** FALSE

5.7.5 Charge-transfer in SAPT

It is possible to obtain the stabilization energy of a complex due to charge-transfer effects from a SAPT computation. The charge-transfer energy can be computed with the sapt module as described by Stone and Misquitta [22].

Charge-transfer energies can be obtained from the following calls to the energy function.

```
energy('sapt0-ct')
energy('sapt2-ct')
energy('sapt2+-ct')
energy('sapt2+(3)-ct')
energy('sapt2+3-ct')
```

A SAPT charge-transfer analysis will perform 5 HF computations: the dimer in the dimer basis, monomer A in the dimer basis, monomer B in the dimer basis, monomer A in the monomer A basis, and monomer B in the monomer B basis. Next, it performs two SAPT computations, one in the dimer basis and one in the monomer basis. Finally, it will print a summary of the charge-transfer results:

```
  SAPT Charge Transfer Analysis
------------------------------------------------------------------------------
  SAPT Induction (Dimer Basis)         -2.0970 mH       -1.3159 kcal mol^-1
  SAPT Induction (Monomer Basis)       -1.1396 mH       -0.7151 kcal mol^-1
  SAPT Charge Transfer                 -0.9574 mH       -0.6008 kcal mol^-1
```

These results are for the water dimer geometry shown above computed with SAPT0/aug-cc-pVDZ.

5.7.6 Interpreting SAPT Results

We will examine the results of a SAPT2+3/aug-cc-pVDZ computation on the water dimer. This computation can be performed with the following input:

```
molecule water_dimer {
     0 1
     O  -1.551007  -0.114520   0.000000
     H  -1.934259   0.762503   0.000000
     H  -0.599677   0.040712   0.000000
     --
     0 1
     O   1.350625   0.111469   0.000000
     H   1.680398  -0.373741  -0.758561
     H   1.680398  -0.373741   0.758561
     units angstrom
     no_reorient
     symmetry c1
}

set globals {
    basis          aug-cc-pvdz
    guess          sad
    scf_type       df
}

set sapt {
    print          1
    nat_orbs       true
    freeze_core    true
}

energy('sapt2+3')
```

To reiterate some of the options mentioned above: the NAT_ORBS option will compute MP2 natural orbitals and use them in the evaluation of the triples correction to dispersion, and the FREEZE_CORE option will freeze the core throughout the SAPT computation. This SAPT2+3/aug-cc-pVDZ computation produces the following results:

```
  SAPT Results
---------------------------------------------------------------------------
  Electrostatics            -13.06429805 mH      -8.19797114 kcal mol^-1
    Elst10,r                -13.37543274 mH      -8.39321111 kcal mol^-1
    Elst12,r                  0.04490253 mH       0.02817676 kcal mol^-1
    Elst13,r                  0.26623216 mH       0.16706321 kcal mol^-1

  Exchange                   13.41793548 mH       8.41988199 kcal mol^-1
    Exch10                   11.21823471 mH       7.03954885 kcal mol^-1
    Exch10(S^2)              11.13803867 mH       6.98922508 kcal mol^-1
    Exch11(S^2)               0.04558910 mH       0.02860760 kcal mol^-1
    Exch12(S^2)               2.15411167 mH       1.35172554 kcal mol^-1

  Induction                  -3.91333155 mH      -2.45565272 kcal mol^-1
    Ind20,r                  -4.57531220 mH      -2.87105187 kcal mol^-1
    Ind30,r                  -4.91715479 mH      -3.08556135 kcal mol^-1
    Ind22                      -0.83761074 mH      -0.52560870 kcal mol^-1
    Exch-Ind20,r                2.47828867 mH       1.55514969 kcal mol^-1
    Exch-Ind30,r                4.33916816 mH       2.72286924 kcal mol^-1
    Exch-Ind22                  0.45370482 mH       0.28470409 kcal mol^-1
    delta HF,r (2)             -1.43240211 mH      -0.89884593 kcal mol^-1
    delta HF,r (3)             -0.85441547 mH      -0.53615383 kcal mol^-1

  Dispersion                   -3.62061213 mH      -2.27196851 kcal mol^-1
    Disp20                     -3.54292109 mH      -2.22321664 kcal mol^-1
    Disp30                      0.05959981 mH       0.03739945 kcal mol^-1
    Disp21                      0.11216179 mH       0.07038259 kcal mol^-1
    Disp22 (SDQ)               -0.17924270 mH      -0.11247650 kcal mol^-1
    Disp22 (T)                 -0.47692549 mH      -0.29927528 kcal mol^-1
    Est. Disp22 (T)            -0.54385253 mH      -0.34127263 kcal mol^-1
    Exch-Disp20                 0.64545652 mH       0.40503010 kcal mol^-1
    Exch-Disp30                -0.01823411 mH      -0.01144207 kcal mol^-1
    Ind-Disp30                 -0.91816995 mH      -0.57616037 kcal mol^-1
    Exch-Ind-Disp30             0.76459013 mH       0.47978757 kcal mol^-1

  Total HF                     -5.68662366 mH      -3.56841037 kcal mol^-1
  Total SAPT0                  -8.58408823 mH      -5.38659691 kcal mol^-1
  Total SAPT2                  -6.72339084 mH      -4.21899163 kcal mol^-1
  Total SAPT2+                 -7.26739725 mH      -4.56036082 kcal mol^-1
  Total SAPT2+(3)              -6.94156528 mH      -4.35589816 kcal mol^-1
  Total SAPT2+3                -7.11337921 mH      -4.46371303 kcal mol^-1
```

At the bottom of this output are the total SAPT energies (defined above), they are composed of subsets of the individual terms printed above. The individual terms are grouped according to the component of the interaction to which they contribute. The total component energies (*i.e.* electrostatics, exchange, induction, and dispersion) represent what we regard as the best estimate available at a given level of SAPT computed from a subset of the terms of that grouping. The groupings shown above are not unique and are certainly not rigorously defined. We regard the groupings used in `PSI4` as a "chemist's grouping" as opposed to a more mathematically based grouping, which would group all exchange terms (*i.e.* $E^{(20)}_{exch-ind,resp}$, $E^{(20)}_{exch-disp}$, *etc.*) in the exchange component. A final note is that both `Disp22(T)` and `Est.Disp22(T)` results appear if MP2 natural orbitals are used to evaluate the triples correction to dispersion. The `Disp22(T)` result is the triples correction as computed in the truncated NO basis; `Est.Disp22(T)` is a scaled result that attempts to recover the effect of the truncated virtual space. The `Est.Disp22(T)` value used in the SAPT energy and dispersion component (see E. G. Hohenstein and C. D. Sherrill, *J. Chem. Phys.*, **133**, 104107 (2010) for details).

5.8 The MRCC Program of M. Kállay

PSI4 contains code to interface to the MRCC program of M. Kállay and J. Gauss. The license and source code of the MRCC program must be obtained from Mihály Kállay (http://www.mrcc.hu/).

5.8.1 Installation

Follow the instructions provided with the source to build the MRCC programs. To be used by PSI4, ensure that the program binary (dmrcc) can be found in your PATH. If PSI4 is unable to execute the binary, an error will be reported.

5.8.2 Running MRCC

MRCC can be invoked in similar fashion as other theories provided in PSI4. For example, if you want to obtain the CCSDT energy for water with cc-pVDZ using MRCC simply provide the following:

```
molecule h2o {
    O
    H 1 1.0
    H 1 1.0 2 104.5
}
set {
    basis cc-pVDZ
}
energy('mrccsdt')
```

mrccsdt in the call to energy() instructs PSI4 to first perform an RHF calculation and then call MRCC to compute the CCSDT energy. For a CCSDT(Q) energy, simply use mrccsdt(q) in the call to energy. MRCC can be used to perform geometry optimization and frequency calculations for electronic ground states only.

At this time, PSI4 is only able to automatically generate the proper input file for MRCC for the methods listed in Table 13. To utilize any method described in Table 13, you must prefix the method name with MR. For other methods, you will be required to use the MRCC keywords described elsewhere in this manual.

Frozen-core approximation is also supported in the MRCC interface. To optimize CH_4 with CCSDT freezing the 1s on carbon, run:

```
molecule H2O {
    O
```

Table 13: Methods available in automatic interface with MRCC.

CCSD	CCSD(T)*	CCSD(T)_L*
CCSDT	CCSDT(Q)*	CCSDT(Q)_L*
CCSDTQ	CCSDTQ(P)*	CCSDTQ(P)_L*
CCSDTQP	CCSDTQP(H)*	CCSDTQP(H)_L*
CCSDTQPH		
CCSDT-1a	CCSDT-1b	CCSDT-3
CCSDTQ-1a	CCSDTQ-1b	CCSDTQ-3
CCSDTQP-1a	CCSDTQP-1b	CCSDTQP-3
CCSDTQPH-1a	CCSDTQPH-1b	CCSDTQPH-3
CC2		
CC3		
CC4		
CC5		
CC6		

*Pertubative methods not available with ROHF reference.

```
    H 1 r
    H 1 r 2 104.5

    r = 1.0
}

set {
    basis cc-pVDZ
    freeze_core true
}

optimize('mrccsdt')
```

5.9 PSIMRCC implementation of Mk-MRCC theory

5.9.1 Introduction to Mk-MRCC

State-specific Multireference coupled cluster theories provide highly accurate energies and properties of electronic states that require a multiconfigurational zeroth-order wavefunction. The PSIMRCC module contained in `PSI4` implements the state-specific multireference coupled-cluster approach of Mukherjee and co-workers (Mk-MRCC). This method is implemented and shown to be a powerful tool in F. A. Evangelista, W. D. Allen, and H. F. Schaefer, III, J. Chem. Phys. 125 (2006) and F. A. Evangelista, A. C. Simmonett, W. D.

Allen, H. F. Schaefer, III, and J. Gauss, J. Chem. Phys. 128, 124104 (2008).
Mk-MRCC is based on the Jeziorski-Monkhorst ansatz [B. Jeziorski and H. J. Monkhorst, Phys. Rev. A 24, 1668 (1981)] for the wavefunction, Ψ

$$|\Psi\rangle = \sum_{\mu}^{d} e^{\hat{T}^{\mu}} |\Phi_{\mu}\rangle c_{\mu} \,,$$

where Φ_{μ} are the reference determinants, $\hat{T}^{\mu}$ are reference-specific excitation operators, and c_{μ} are expansion coefficients obtained through diagonalization of the Mk-MRCC effective Hamiltonian matrix that allows the various reference determinants to interact. As an example of how this works the Mk-MRCCSD excitation operators for each reference is contracted two-body terms

$$\hat{T}^{\mu} = \hat{T}_1^{\mu} + \hat{T}_2^{\mu}$$

where

$$\hat{T}_1^{\mu} = \sum_{i}^{\text{occ}(\mu)} \sum_{a}^{\text{vir}(\mu)} t_i^a(\mu) \hat{a}_a^{\dagger} \hat{a}_i$$

and

$$\hat{T}_2^{\mu} = \frac{1}{4} \sum_{i}^{\text{occ}(\mu)} \sum_{a}^{\text{vir}(\mu)} t_{ij}^{ab}(\mu) \hat{a}_b^{\dagger} \hat{a}_j \hat{a}_a^{\dagger} \hat{a}_i$$

The Mk-MRCC energy is a chosen eigenvalue of the effective Hamiltonian, $\mathrm{H}_{\mu\nu}^{eff}$

$$\sum_{\nu} \mathrm{H}_{\mu\nu}^{eff} c_{\nu} = E c_{\nu}$$

where

$$\mathrm{H}_{\mu\nu}^{eff} = \langle \Phi_{\mu} | \hat{H} e^{\hat{T}^{\nu}} | \Phi_{\nu} \rangle \ .$$

`PSI4` currently has Mk-MRCC with singles and doubles [Mk-MRCCSD], single and doubles with perturbative triples[Mk-MRCCSD(T)]. A companion perturbation method(Mk-MRPT2) has been developed based on the Mukherjee formalisim as shown in F. A. Evangelista, A. C. Simmonett, H. F. Schaefer, III, D. Mukherjee, and W. D. Allen, Phys. Chem. Chem. Phys. 11, 4728 (2009).

5.9.2 Orbital ordering and selection of the model space

The reference determinants Φ_{μ} are specified in PSIMRCC via occupational numbers. PSIMRCC requires that four arrays be specified for this purpose.

- Frozen doubly occupied orbitals (`FROZEN_DOCC`) are doubly occupied in each reference determinant and are not correlated in the MRCC procedure.
- Doubly occupied orbitals(`RESTRICTED_DOCC`) are doubly occupied in each reference determinant and are correlated in the MRCC procedure.

- Active orbitals(`ACTIVE`) are partially occupied in each reference determinant.
- Frozen virtual orbitals (`FROZEN_UOCC`) are unoccupied in all reference determinants and are excluded from the correlated wave function.

The model space is selected by considering all possible occupations of the electrons among the orbitals in the active space that result in determinants with the correct symmetry (`WFN_SYM`) and the correct M_s value specified by the keyword MULTP. Note that this does not consider the multiplicity of the wavefunction. Thus, in order to obtain the wavefunction with a set of $M_s = 0$ reference determinants for an open-shell system you should request a MULTP of 1 within the PSIMRCC module, and select the root of the effective Hamiltonian that corresponds to the state of interest. In addition, the `WFN_SYM` keyword needs to be specified otherwise the wavefunction belonging to the all-symmetric irrep will be selected. In addition, it should be noted that for an open-shell singlet based on two $M_s = 0$ determinants the eigenvector is $[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}]$, which corresponds to a wavefunction of the following form:

$$\frac{1}{\sqrt{2}}\left(\chi_1\alpha(1)\chi_2\beta(2) + \chi_2\alpha(1)\chi_1\beta(2)\right)$$

6 Geometry Optimization

`PSI4` carries out optimizations using a module called `optking`. The optking program performs optimization of molecular geometries given nuclear gradients, and optionally second derivatives in cartesian coordinates. The default minimization algorithm employs redundant internal coordinates with an RFO step and a BFGS Hessian update.

The principal literature references include the introduction of redundant internal coordinates in C. Peng, P.Y. Ayala, H.B. Schlegel, and M.J. Frisch, J. Comput. Chem., 17, 49 (1996). The general approach employed in this code is similar to the 'model Hessian plus RF method' described and tested in V. Bakken and T. Helgaker, J. Chem. Phys., 117, 9160 (2002). (However, for separated fragments, we have chosen not to employ their 'extra-redundant' coordinates defined by their 'auxiliary interfragment' bonds.)

6.1 Hessian

If cartesian second derivatives have been computed, optking can read them and transform them into internal coordinates to make the initial Hessian. Otherwise, an initial empirical Hessian is determined according to either Schlegel, Theor. Chim. Acta, 66, 333 (1984) or Fischer and Almlof, J. Phys. Chem., 96, 9770 (1992). For interfragment coordinates, the initial guess can be computed using the Fischer formulas or a simple diagonal guess. For Hessian update formulas, see Schlegel Ab Initio Methods in Quantum Chemistry (1987) and J. M. Bofill, J. Comp. Chem., Vol. 15, pages 1-11 (1994).

The Hessian may be updated every few iterations using the following keyword:

`FULL_HESS_EVERY`	Frequency with which to compute the full Hessian in the course of a geometry optimization. 0 means to compute the initial Hessian only, 1 means recompute every step, and N means recompute every N steps. The default (-1) is to never compute the full Hessian. **Type:** integer **Default:** -1

6.2 Convergence Criteria

`optking` monitors five quantities to evaluate the progress of a geometry optimization. These are (with their keywords) the change in energy (`MAX_ENERGY_G_CONVERGENCE`), the maximum element of the gradient (`MAX_FORCE_G_CONVERGENCE`), the root-mean-square of the gradient (`RMS_FORCE_G_CONVERGENCE`), the maximum element of displacement (`MAX_DISP_G_CONVERGENCE`), and the root-mean-square of displacement (`RMS_DISP_G_CONVERGENCE`), all in internal coordinates and atomic units. Usually, these options will not be set directly. Primary control for geometry convergence lies with the keyword `G_CONVERGENCE` which sets the aforementioned in accordance with Table 14.

Table 14: Summary of sets of geometry optimization criteria available in `PSI4`.

`G_CONVERGENCE`	Max Energy	Max Force	RMS Force	Max Disp	RMS Disp	Notes
NWCHEM_LOOSE		4.5×10^{-3}	3.0×10^{-3}	5.4×10^{-3}	3.6×10^{-3}	*d*
GAU_LOOSE		2.5×10^{-3}	1.7×10^{-3}	1.0×10^{-2}	6.7×10^{-3}	*f*
TURBOMOLE	1.0×10^{-6}	1.0×10^{-3}	5.0×10^{-4}	1.0×10^{-3}	5.0×10^{-4}	*d*
GAU		4.5×10^{-4}	3.0×10^{-4}	1.8×10^{-3}	1.2×10^{-3}	*c*, *f*
CFOUR			1.0×10^{-4}			*d*
QCHEM	1.0×10^{-6}	3.0×10^{-4}		1.2×10^{-3}		*a*, *e*
MOLPRO	1.0×10^{-6}	3.0×10^{-4}		3.0×10^{-4}		*b*, *e*
GAU_TIGHT		1.5×10^{-5}	1.0×10^{-5}	6.0×10^{-5}	4.0×10^{-5}	*c*, *f*
GAU_VERYTIGHT		2.0×10^{-6}	1.0×10^{-6}	6.0×10^{-6}	4.0×10^{-6}	*f*

[a] Default

[b] Baker convergence criteria are the same.

[c] Counterpart NWCHEM convergence criteria are the same.

[d] Convergence achieved when all active criteria are fulfilled.

[e] Convergence achieved when MAX_FORCE and one of MAX_ENERGY or MAX_DISP are fulfilled.

[f] Normal convergence achieved when all four criteria are fulfilled. To help with flat potential surfaces, alternate convergence achieved when 100× rms force is less than RMS_FORCE criterion.

For ultimate control, specifying a value for any of the five monitored options activates that criterium and overwrites/appends it to the criteria set by `G_CONVERGENCE`. Note that this revokes the special convergence arrangements detailed in notes *e* and *f* in Table 14 and instead requires all active criteria to be fulfilled to achieve convergence.

The progress of a geometry optimization can be monitored by grepping `output.dat` for the tilde character (~). This produces a table like the one below that shows for each iteration the value for each of the five quantities and whether the criterion is active and fulfilled (*), active and unfulfilled ( ), or inactive (o).

```
--------------------------------------------------------------------------------------------- ~
 Step     Total Energy     Delta E     MAX Force     RMS Force      MAX Disp      RMS Disp    ~
--------------------------------------------------------------------------------------------- ~
  Convergence Criteria    1.00e-06 *    3.00e-04 *             o    1.20e-03 *             o  ~
--------------------------------------------------------------------------------------------- ~
    1     -38.91591820   -3.89e+01      6.91e-02      5.72e-02 o    1.42e-01      1.19e-01 o  ~
    2     -38.92529543   -9.38e-03      6.21e-03      3.91e-03 o    2.00e-02      1.18e-02 o  ~
    3     -38.92540669   -1.11e-04      4.04e-03      2.46e-03 o    3.63e-02      2.12e-02 o  ~
    4     -38.92548668   -8.00e-05      2.30e-04 *    1.92e-04 o    1.99e-03      1.17e-03 o  ~
    5     -38.92548698   -2.98e-07 *    3.95e-05 *    3.35e-05 o    1.37e-04 *    1.05e-04 o  ~
```

The full list of keywords for `optking` is provided in Appendix B.25.

7 Additional Documentation

Additional information and the most recent version of this manual may be found at the `PSI` website `www.psicode.org`. More complete information on the installation of the `PSI4` package is available in the PSI4 Installation Manual. For programmers, the PSI4 Programmer's Manual is available online at the `PSI` development Trac page.

References

[1] B. Jeziorski, R. Moszynski, and K. Szalewicz, Chem. Rev. **94**, 1887 (1994).

[2] E. G. Hohenstein and C. D. Sherrill, Wiley Interdisciplinary Reviews: Comput. Mol. Sci. **X**, XX (2012).

[3] E. Papajak and D. G. Truhlar, J. Chem. Theory Comput. **7**, 10 (2011).

[4] E. G. Hohenstein and C. D. Sherrill, J. Chem. Phys. **132**, 184111 (2010).

[5] E. G. Hohenstein and C. D. Sherrill, J. Chem. Phys. **133**, 014101 (2010).

[6] E. Cohen, T. Cvitas, J. Frey, B. Holmström, K. Kuchitsu, R. Marquardt, I. Mills, F. Pavese, M. Quack, J. Stohner, H. Strauss, M. Takami, and A. Thor, *Quantities, Units and Symbols in Physical Chemistry. IUPAC Green Book, 3rd Edition* (IUPAC & RSC Publishing, Cambridge, England, 2008).

[7] http://pubchem.ncbi.nlm.nih.gov.

[8] T. Tsuchimochi and G. E. Scuseria, J. Chem. Phys. **133**, 141102 (2010).

[9] J. A. Pople, P. M. W. Gill, and N. C. Handy, Int. J. Quantum Chem. **56**, 303 (1995).

[10] C. D. Sherrill and H. F. Schaefer, in *Advances in Quantum Chemistry*, edited by P.-O. Löwdin (Academic Press, New York, 1999), Vol. 34, pp. 143–269.

[11] N. C. Handy, Chem. Phys. Lett. **74**, 280 (1980).

[12] J. Olsen, B. O. Roos, P. Jørgensen, and H. J. Aa. Jensen, J. Chem. Phys. **89**, 2185 (1988).

[13] S. Hirata and R. J. Bartlett, Chem. Phys. Lett. **321**, 216 (2000).

[14] J. Olsen, J. Chem. Phys. **113**, 7140 (2000).

[15] P. Piecuch, S. A. Kucharski, and R. J. Bartlett, J. Chem. Phys. **110**, 6103 (1999).

[16] G. D. Purvis and R. J. Bartlett, J. Chem. Phys. **76**, 1910 (1982).

[17] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, Chem. Phys. Lett. **157**, 479 (1989).

[18] J. Schirmer, Phys. Rev. A **26**, 2395 (1982).

[19] A. B. Trofimov, I. L. Krivdina, J. Weller, and J. Schirmer, Chem. Phys. **329**, 1 (2006).

[20] C. Hättig and K. Hald, Phys. Chem. Chem. Phys. **4**, 2111 (2002).

[21] M. Saitow and Y. Mochizuki, Chem. Phys. Lett. **X**, XX (2012).

[22] A. J. Stone and A. J. Misquitta, Chem. Phys. Lett. **473**, 201 (2009).

A `PSI4` Reference

J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer, R. A. King, E. F. Valeev, C. D. Sherrill, and T. D. Crawford, *WIREs: Comput. Mol. Sci.*, in press (doi: 10.1002/wcms.93).

B Keywords Recognized by Each Module

This listing provides details of the keywords recognized by each module in the code and, where appropriate, their allowed values. Certain convergence and tolerance keywords, of type *double* (real numbers), may be specified using either a real number or an integer; an integer X is then treated as the number of converged decimal digits required. For example, to request an energy converged to $10^{-6}E_h$, the user may set the `e_convergence` keyword to 0.000001, 1.0e-6, or 6.

B.1 GLOBALS

BENCH

Some codes (DFT) can dump benchmarking data to separate output files
Type: integer **Default:** 0

DOCC

An array containing the number of doubly-occupied orbitals per irrep (in Cotton order)
Type: array **Default:** No Default

FREEZE_CORE

Specifies how many core orbitals to freeze in correlated computations. TRUE will default to freezing the standard default number of core orbitals. For heavier elements, there can be some ambiguity in how many core orbitals to freeze; in such cases, SMALL picks the most conservative standard setting (freezes fewer orbitals), and LARGE picks the least conservative standard setting (freezes more orbitals). More precise control over the number of frozen orbitals can be attained by using the keywords NUM_FROZEN_DOCC (gives the total number of orbitals to freeze, program picks the lowest-energy orbitals) or FROZEN_DOCC (gives the number of orbitals to freeze per irreducible representation)
Possible Values: FALSE, TRUE, SMALL, LARGE
Type: string **Default:** FALSE

FROZEN_DOCC

An array containing the number of frozen doubly-occupied orbitals per irrep (these are not excited in a correlated wavefunction, nor can they be optimized in MCSCF
Type: array **Default:** No Default

FROZEN_UOCC

An array containing the number of frozen unoccupied orbitals per irrep (these are not populated in a correlated wavefunction, nor can they be optimized in MCSCF
Type: array **Default:** No Default

FULL_HESS_EVERY

Frequency with which to compute the full Hessian in the course of a geometry optimization. 0 means to compute the initial Hessian only, 1 means recompute every step, and N means recompute every N steps. The default (-1) is to never compute the full Hessian.
Type: integer **Default:** -1

GEOM_MAXITER

Maximum number of geometry optimization steps
Type: integer **Default:** 20

NUM_FROZEN_DOCC

The number of core orbitals to freeze in later correlated computations. FROZEN_DOCC trumps this option

Type: integer **Default:** 0

NUM_FROZEN_UOCC

The number of virtual orbitals to freeze in later correlated computations. FROZEN_UOCC trumps this option

Type: integer **Default:** 0

PRINT

The amount of information to print to the output file. 1 prints basic information, and higher levels print more information. A value of 5 will print very large amounts of debugging information.

Type: integer **Default:** 1

PUREAM

Do use pure angular momentum basis functions? If not explicitly set, the default comes from the basis set.

Type: boolean **Default:** true

SOCC

An array containing the number of singly-occupied orbitals per irrep (in Cotton order)

Type: array **Default:** No Default

UNITS

Units used in geometry specification
Possible Values: BOHR, AU, A.U., ANGSTROMS, ANG, ANGSTROM

Type: string **Default:** ANGSTROMS

B.2 ADC

Performs Algebraic-Diagrammatic Construction (ADC) propagator computations for excited states.

CACHELEVEL

How to cache quantities within the DPD library

Type: integer **Default:** 2

MEMORY

The amount of memory available (in Mb)

Type: integer **Default:** 1000

NEWTON_CONVERGENCE

The convergence criterion for pole searching step. See the note at the beginning of Section B.

Type: double **Default:** 1e-7

NORM_TOLERANCE

The cutoff norm of residual vector in SEM step. See the note at the beginning of Section B.

Type: double **Default:** 1e-6

NUM_AMPS_PRINT

Number of components of transition amplitudes printed

Type: integer **Default:** 5

POLE_MAXITER

Maximum iteration number in pole searching

Type: integer **Default:** 20

PR

Do use the partial renormalization scheme for the ground state wavefunction?

Type: boolean **Default:** false

REFERENCE

The Reference

Type: string **Default:** No Default

SEM_MAXITER

Maximum iteration number in simultaneous expansion method

Type: integer **Default:** 30

STATES_PER_IRREP

The poles per irrep vector

Type: array **Default:** No Default

B.3 CCDENSITY

Computes the coupled cluster density matrices. Called whenever CC properties and/or gradients are required.

AO_BASIS

The algorithm to use for the $\langle VV||VV \rangle$ terms
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

CACHELEVEL

The amount of cacheing of data to perform
Type: integer **Default:** 2

GAUGE

The type of gauge to use for properties
Type: string **Default:** LENGTH

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.
Type: double **Default:** 1e-14

OPDM_RELAX

Do relax the one-particle density matrix?
Type: boolean **Default:** false

PROP_ALL

Do compute all relaxed excited states?
Type: boolean **Default:** false

PROP_ROOT

Type: integer **Default:** 1

PROP_SYM

The symmetry of states

Type: integer **Default:** 1

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

STATES_PER_IRREP

The number of electronic states to computed, per irreducible representation

Type: array **Default:** No Default

XI

Do compute Xi?

Type: boolean **Default:** false

ZETA

Do ?

Type: boolean **Default:** false

B.4 CCENERGY

Computes coupled cluster energies. Called as part of any coupled cluster computation.

ABCD

Possible Values: NEW, OLD

Type: string **Default:** NEW

ANALYZE

Do ?

Type: boolean **Default:** false

BRUECKNER_ORBS_R_CONVERGENCE

Convergence criterion for Breuckner orbitals. The convergence is determined based on the largest T_1 amplitude. See the note at the beginning of Section B.

Type: double **Default:** 1e-5

CACHELEVEL

Cacheing level for libdpd governing the storage of amplitudes, integrals, and intermediates in the CC procedure. A value of 0 retains no quantities in cache, while a level of 6 attempts to store all quantities in cache. For particularly large calculations, a value of 0 may help with certain types of memory problems. The default is 2, which means that all four-index quantites with up to two virtual-orbital indices (e.g., $\langle ij|ab\rangle >$ integrals) may be held in the cache.

Type: integer **Default:** 2

CACHETYPE

Selects the priority type for maintaining the automatic memory cache used by the libdpd codes. A value of LOW selects a ”low priority” scheme in which the deletion of items from the cache is based on pre-programmed priorities. A value of LRU selects a ”least recently used” scheme in which the oldest item in the cache will be the first one deleted.
Possible Values: LOW, LRU

Type: string **Default:** LOW

CC_OS_SCALE

Type: double **Default:** 1.27

CC_SS_SCALE

Type: double **Default:** 1.13

DIIS

Do use DIIS extrapolation to accelerate convergence?

Type: boolean **Default:** true

LOCAL

Do simulate the effects of local correlation techniques?

Type: boolean **Default:** false

LOCAL_CUTOFF

Value (always between one and zero) for the Broughton-Pulay completeness check used to contruct orbital domains for local-CC calculations. See J. Broughton and P. Pulay, J. Comp. Chem. 14, 736-740 (1993) and C. Hampel and H.-J. Werner, J. Chem. Phys. 104, 6286-6297 (1996).

Type: double **Default:** 0.02

LOCAL_METHOD

Type of local-CCSD scheme to be simulated. WERNER selects the method developed by H.-J. Werner and co-workers, and AOBASIS selects the method developed by G.E. Scuseria and co-workers (currently inoperative).

Possible Values: WERNER, AOBASIS

Type: string **Default:** WERNER

LOCAL_PAIRDEF

Possible Values: BP, RESPONSE

Type: string **Default:** BP

LOCAL_WEAKP

Desired treatment of "weak pairs" in the local-CCSD method. A value of NEGLECT ignores weak pairs entirely. A value of NONE treats weak pairs in the same manner as strong pairs. A value of MP2 uses second-order perturbation theory to correct the local-CCSD energy computed with weak pairs ignored.

Possible Values: NONE, NEGLECT, MP2

Type: string **Default:** NONE

MAXITER

Maximum number of iterations to solve the CC equations

Type: integer **Default:** 50

MP2_AMPS_PRINT

Do print the MP2 amplitudes which are the starting guesses for RHF and UHF reference functions?

Type: boolean **Default:** false

MP2_OS_SCALE

Type: double **Default:** 1.20

MP2_SS_SCALE

Type: double **Default:** 1.0/3.0

NEW_TRIPLES

Do ?

Type: boolean **Default:** true

NUM_AMPS_PRINT

Number of important t_1 and t_2 amplitudes to print

Type: integer **Default:** 10

NUM_THREADS

Number of threads

Type: integer **Default:** 1

PAIR_ENERGIES_PRINT

Do print MP2 and CCSD pair energies for RHF references?

Type: boolean **Default:** false

PROPERTY

Possible Values: POLARIZABILITY, ROTATION, MAGNETIZABILITY, ROA, ALL

Type: string **Default:** POLARIZABILITY

REFERENCE

Reference wavefunction type

Possible Values: RHF, ROHF, UHF

Type: string **Default:** RHF

RESTART

Do restart the coupled-cluster iterations from old t_1 and t_2 amplitudes? For geometry optimizations, Brueckner calculations, etc. the iterative solution of the CC amplitude equations may benefit considerably by reusing old vectors as initial guesses. Assuming that the MO phases remain the same between updates, the CC codes will, by default, re-use old vectors, unless the user sets RESTART = false.

Type: boolean **Default:** true

R_CONVERGENCE

Convergence criterion for wavefunction (change) in CC amplitude equations. See the note at the beginning of Section B.

Type: double **Default:** 1e-7

SCSN_MP2

Do ?

Type: boolean **Default:** false

SCS_CCSD

Do ?

Type: boolean **Default:** false

SCS_MP2

Do ?

Type: boolean **Default:** false

SEMICANONICAL

Convert ROHF MOs to semicanonical MOs

Type: boolean **Default:** true

SPINADAPT_ENERGIES

Do print spin-adapted pair energies?

Type: boolean **Default:** false

T2_COUPLED

Do ?

Type: boolean **Default:** false

T3_WS_INCORE

Do ?

Type: boolean **Default:** false

B.5 CCEOM

Performs equation-of-motion (EOM) coupled cluster excited state computations.

ABCD

Possible Values: NEW, OLD
Type: string **Default:** NEW

CACHELEVEL

Type: integer **Default:** 2

CACHETYPE

Possible Values: LOW, LRU
Type: string **Default:** LRU

CC3_FOLLOW_ROOT

Do ?
Type: boolean **Default:** false

COLLAPSE_WITH_LAST

Do ?
Type: boolean **Default:** true

COMPLEX_TOLERANCE

See the note at the beginning of Section B.
Type: double **Default:** 1e-12

EOM_GUESS

Specifies a set of single-excitation guess vectors for the EOM-CC procedure. If EOM_GUESS = SINGLES, the guess will be taken from the singles-singles block of the similarity-transformed Hamiltonian, Hbar. If EOM_GUESS = DISK, guess vectors from a previous computation will be read from disk. If EOM_GUESS = INPUT, guess vectors will be specified in user input. The latter method is not currently available.
Possible Values: SINGLES, DISK, INPUT
Type: string **Default:** SINGLES

EOM_REFERENCE

Possible Values: RHF, ROHF, UHF

Type: string **Default:** RHF

E_CONVERGENCE

Convergence criterion for excitation energy (change) in the Davidson algorithm for CC-EOM. See the note at the beginning of Section B.

Type: double **Default:** 1e-8

FULL_MATRIX

Do ?

Type: boolean **Default:** false

LOCAL

Do ?

Type: boolean **Default:** false

LOCAL_CUTOFF

Type: double **Default:** 0.02

LOCAL_DO_SINGLES

Do ?

Type: boolean **Default:** true

LOCAL_FILTER_SINGLES

Do ?

Type: boolean **Default:** true

LOCAL_GHOST

Type: integer **Default:** -1

LOCAL_METHOD

Possible Values: WERNER, AOBASIS
Type: string **Default:** WERNER

LOCAL_PRECONDITIONER

Possible Values: HBAR, FOCK
Type: string **Default:** HBAR

LOCAL_WEAKP

Possible Values: NONE, MP2, NEGLECT
Type: string **Default:** NONE

MAXITER

Maximum number of iterations
Type: integer **Default:** 80

NEW_TRIPLES

Do ?
Type: boolean **Default:** true

NUM_AMPS_PRINT

Number of important CC amplitudes to print
Type: integer **Default:** 5

NUM_THREADS

Number of threads
Type: integer **Default:** 1

PROP_ROOT

Root number (within its irrep) for computing properties. Defaults to highest root requested.
Type: integer **Default:** 0

PROP_SYM

Symmetry of the state to compute properties. Defaults to last irrep for which states are requested.
Type: integer **Default:** 1

REFERENCE

Reference wavefunction type
Possible Values: RHF, ROHF, UHF
Type: string **Default:** RHF

RESTART_EOM_CC3

Do ?
Type: boolean **Default:** false

RHF_TRIPLETS

Do ?
Type: boolean **Default:** false

R_CONVERGENCE

Convergence criterion for norm of the residual vector in the Davidson algorithm for CC-EOM. See the note at the beginning of Section B.
Type: double **Default:** 1e-6

SCHMIDT_ADD_RESIDUAL_TOLERANCE

Minimum absolute value above which a guess vector to a root is added to the Davidson algorithm in the EOM-CC iterative procedure. See the note at the beginning of Section B.
Type: double **Default:** 1e-3

SEMICANONICAL

Convert ROHF MOs to semicanonical MOs
Type: boolean **Default:** true

SINGLES_PRINT

Do print information on the iterative solution to the single-excitation EOM-CC problem used as a guess to full EOM-CC?
Type: boolean **Default:** false

SS_E_CONVERGENCE

Convergence criterion for excitation energy (change) in the Davidson algorithm for the CIS guess to CC-EOM. See the note at the beginning of Section B.

Type: double **Default:** 1e-6

SS_R_CONVERGENCE

Convergence criterion for norm of the residual vector in the Davidson algorithm for the CIS guess to CC-EOM. See the note at the beginning of Section B.

Type: double **Default:** 1e-6

SS_SKIP_DIAG

Do ?

Type: boolean **Default:** false

SS_VECS_PER_ROOT

Type: integer **Default:** 5

STATES_PER_IRREP

Number of excited states per irreducible representation for EOM-CC and CC-LR calculations. Irreps denote the final state symmetry, not the symmetry of the transtion.

Type: array **Default:** No Default

T3_WS_INCORE

Do ?

Type: boolean **Default:** false

VECS_CC3

Type: integer **Default:** 10

VECS_PER_ROOT

Type: integer **Default:** 12

B.6 CCHBAR

Assembles the coupled cluster effective Hamiltonian. Called whenever CC properties and/or gradients are required.

CACHELEVEL

Type: integer **Default:** 2

EOM_REFERENCE

Type: string **Default:** RHF

T_AMPS

Do compute the Tamplitude equation matrix elements?
Type: boolean **Default:** false

WABEI_LOWDISK

Do use the minimal-disk algorithm for Wabei? It's VERY slow!
Type: boolean **Default:** false

B.7 CCLAMBDA

Solves for the Lagrange multipliers, which are needed whenever coupled cluster properties or gradients are requested.

ABCD

Type: string **Default:** NEW

AO_BASIS

The algorithm to use for the $\langle VV||VV \rangle$ terms
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

CACHELEVEL

Type: integer **Default:** 2

DIIS

Do use DIIS extrapolation to accelerate convergence?

Type: boolean **Default:** true

LOCAL

Do ?

Type: boolean **Default:** false

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0.10

LOCAL_CUTOFF

Type: double **Default:** 0.02

LOCAL_FILTER_SINGLES

Do ?

Type: boolean **Default:** true

LOCAL_METHOD

Type: string **Default:** WERNER

LOCAL_PAIRDEF

Type: string **Default:** No Default

LOCAL_WEAKP

Type: string **Default:** NONE

MAXITER

Maximum number of iterations

Type: integer **Default:** 50

NUM_AMPS_PRINT

Type: integer **Default:** 10

PROP_ALL

Do ?

Type: boolean **Default:** false

PROP_ROOT

Type: integer **Default:** 1

PROP_SYM

Type: integer **Default:** 1

RESTART

Do ?

Type: boolean **Default:** false

R_CONVERGENCE

Convergence criterion for wavefunction (change) in CC lambda-amplitude equations. See the note at the beginning of Section B.

Type: double **Default:** 1e-7

SEKINO

Do ?

Type: boolean **Default:** false

STATES_PER_IRREP

Type: array **Default:** No Default

ZETA

Type: boolean **Default:** false

B.8 CCRESPONSE

Performs coupled cluster response property computations.

ABCD

Type: string **Default:** NEW

ANALYZE

Do ?

Type: boolean **Default:** false

CACHELEVEL

Cacheing level for libdpd

Type: integer **Default:** 2

DIIS

Do use DIIS extrapolation to accelerate convergence?

Type: boolean **Default:** true

GAUGE

Gauge for optical rotation

Type: string **Default:** LENGTH

LINEAR

Do Bartlett size-extensive linear model?

Type: boolean **Default:** false

LOCAL

Do simulate local correlation?

Type: boolean **Default:** false

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0.10

LOCAL_CUTOFF

Type: double **Default:** 0.01

LOCAL_FILTER_SINGLES

Do ?

Type: boolean **Default:** false

LOCAL_METHOD

Type: string **Default:** WERNER

LOCAL_PAIRDEF

Type: string **Default:** NONE

LOCAL_WEAKP

Type: string **Default:** NONE

MAXITER

Maximum number of iterations to converge perturbed amplitude equations

Type: integer **Default:** 50

NUM_AMPS_PRINT

Type: integer **Default:** 5

OMEGA

Array that specifies the desired frequencies of the incident radiation field in CCLR calculations. If only one element is given, the units will be assumed to be atomic units. If more than one element is given, then the units must be specified as the final element of the array. Acceptable units are HZ, NM, EV, and AU.

Type: array **Default:** No Default

PROPERTY

The response property desired. Acceptable values are POLARIZABILITY (default) for dipole-polarizabilities, ROTATION for specific rotations, ROA for Raman Optical Activity, and ALL for all of the above.

Possible Values: POLARIZABILITY, ROTATION, ROA, ALL

Type: string **Default:** POLARIZABILITY

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

RESTART

Do restart from on-disk amplitudes?

Type: boolean **Default:** true

R_CONVERGENCE

Convergence criterion for wavefunction (change) in perturbed CC equations. See the note at the beginning of Section B.

Type: double **Default:** 1e-7

SEKINO

Do Sekino-Bartlett size-extensive model-III?

Type: boolean **Default:** false

B.9 CCSORT

Sorts integrals for efficiency. Called before (non density-fitted) MP2 and coupled cluster computations.

AO_BASIS

The algorithm to use for the $\langle VV||VV\rangle$ terms
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

CACHELEVEL

Type: integer **Default:** 2

EOM_REFERENCE

Type: string **Default:** RHF

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.
Type: double **Default:** 1e-14

KEEP_OEIFILE

Do retain the input one-electron integrals?
Type: boolean **Default:** false

KEEP_TEIFILE

Do retain the input two-electron integrals?
Type: boolean **Default:** false

LOCAL

Do ?
Type: boolean **Default:** false

LOCAL_CORE_CUTOFF

Type: double **Default:** 0.05

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0.10

LOCAL_CUTOFF

Type: double **Default:** 0.02

LOCAL_DOMAIN_MAG

Do ?

Type: boolean **Default:** false

LOCAL_DOMAIN_POLAR

Do ?

Type: boolean **Default:** false

LOCAL_DOMAIN_SEP

Do ?

Type: boolean **Default:** false

LOCAL_FILTER_SINGLES

Do ?

Type: boolean **Default:** false

LOCAL_METHOD

Type: string **Default:** WERNER

LOCAL_PAIRDEF

Type: string **Default:** BP

LOCAL_WEAKP

Type: string **Default:** NONE

OMEGA

Energy of applied field [au] for dynamic properties

Type: array **Default:** No Default

PROPERTY

Type: string **Default:** POLARIZABILITY

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

SEMICANONICAL

Convert ROHF MOs to semicanonical MOs

Type: boolean **Default:** true

B.10 CCTRIPLES

Computes the triples component of CCSD(T) energies (and gradients, if necessary).

NUM_THREADS

Number of threads

Type: integer **Default:** 1

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

SEMICANONICAL

Convert ROHF MOs to semicanonical MOs

Type: boolean **Default:** true

B.11 CIS

Performs configuration interaction singles (CIS) computations. Currently unused in Psi4.

DIAG_METHOD

Possible Values: DAVIDSON, FULL
Type: string **Default:** DAVIDSON

DOMAINS

Type: array **Default:** No Default

DOMAIN_PRINT

Do ?
Type: boolean **Default:** false

LOCAL

Do ?
Type: boolean **Default:** false

LOCAL_AMPS_PRINT_CUTOFF

Type: double **Default:** 0.60

LOCAL_CUTOFF

Type: double **Default:** 0.02

LOCAL_GHOST

Type: integer **Default:** -1

LOCAL_METHOD

Possible Values: AOBASIS, WERNER
Type: string **Default:** WERNER

LOCAL_WEAKP

Possible Values: MP2, NEGLECT, NONE
Type: string **Default:** MP2

MAXITER

Maximum number of iterations
Type: integer **Default:** 500

REFERENCE

Reference wavefunction type
Possible Values: RHF, ROHF, UHF
Type: string **Default:** RHF

R_CONVERGENCE

Convergence criterion for CIS wavefunction. See the note at the beginning of Section B.
Type: double **Default:** 1e-7

STATES_PER_IRREP

The number of electronic states to computed, per irreducible representation
Type: array **Default:** No Default

B.12 CLAG

Solves for the CI Lagrangian. Called whenver CI properties or gradients are requested.

CAS_FILES_WRITE

Do write the OEI, TEI, OPDM, TPDM, and Lagrangian files in canonical form, Pitzer order?
Type: boolean **Default:** false

FOLLOW_ROOT

Root to get OPDM
Type: integer **Default:** 1

B.13 CPHF

CIS_AD_STATES

Which states to save AD Matrices for? * Positive - Singlets * Negative - Triplets *

Type: array **Default:** No Default

CIS_AMPLITUDE_CUTOFF

Minimum singles amplitude to print in CIS analysis

Type: double **Default:** 0.15

CIS_DOPDM_STATES

Which states to save AO difference OPDMs for? * Positive - Singlets * Negative - Triplets *

Type: array **Default:** No Default

CIS_MEM_SAFETY_FACTOR

Memory safety factor for allocating JK

Type: double **Default:** 0.75

CIS_NO_STATES

Which states to save AO Natural Orbitals for? * Positive - Singlets * Negative - Triplets *

Type: array **Default:** No Default

CIS_OPDM_STATES

Which states to save AO OPDMs for? * Positive - Singlets * Negative - Triplets *

Type: array **Default:** No Default

CIS_TOPDM_STATES

Which states to save AO transition OPDMs for? * Positive - Singlets * Negative - Triplets *

Type: array **Default:** No Default

CPHF_MEM_SAFETY_FACTOR

Memory safety factor for allocating JK

Type: double **Default:** 0.75

CPHF_TASKS

Which tasks to run CPHF For * Valid choices: * -Polarizability *

Type: array **Default:** No Default

DEBUG

The amount of debug information printed to the output file

Type: integer **Default:** 0

DO_SINGLETS

Do singlet states? Default true

Type: boolean **Default:** true

DO_TRIPLETS

Do triplet states? Default true

Type: boolean **Default:** true

EXPLICIT_HAMILTONIAN

Do explicit hamiltonian only?

Type: boolean **Default:** false

FITTING_ALGORITHM

Fitting algorithm (0 for old, 1 for new)

Type: integer **Default:** 0

FITTING_CONDITION

The maximum reciprocal condition allowed in the fitting metric

Type: double **Default:** 1.0e-12

MODULE

What app to test?
Possible Values: RCIS, RCPHF, RTDHF, RCPKS, RTDA, RTDDFT

Type: string **Default:** RCIS

OMP_N_THREAD

The maximum number of integral threads (0 for omp_get_max_threads())

Type: integer **Default:** 0

PRINT

The amount of information printed to the output file

Type: integer **Default:** 1

RI_BASIS_SCF

Auxiliary basis for SCF

Type: string **Default:** No Default

SCF_TYPE

SCF Type
Possible Values: DIRECT, DF, PK, OUT_OF_CORE, PS

Type: string **Default:** DIRECT

SCHWARZ_CUTOFF

The schwarz cutoff value

Type: double **Default:** 1.0e-12

SOLVER_CONVERGENCE

Solver convergence threshold (max 2-norm). See the note at the beginning of Section B.

Type: double **Default:** 1.0e-6

SOLVER_EXACT_DIAGONAL

Solver exact diagonal or eigenvalue difference?

Type: boolean **Default:** false

SOLVER_MAXITER

Solver maximum iterations

Type: integer **Default:** 100

SOLVER_MAX_SUBSPACE

DL Solver maximum number of subspace vectors
Type: integer **Default:** 6

SOLVER_MIN_SUBSPACE

DL Solver number of subspace vectors to collapse to
Type: integer **Default:** 2

SOLVER_NORM

DL Solver minimum corrector norm to add to subspace
Type: double **Default:** 1.0e-6

SOLVER_N_GUESS

DL Solver number of guesses
Type: integer **Default:** 1

SOLVER_N_ROOT

DL Solver number of roots
Type: integer **Default:** 1

SOLVER_PRECONDITION

Solver precondition type
Possible Values: SUBSPACE, JACOBI, NONE
Type: string **Default:** JACOBI

SOLVER_PRECONDITION_MAXITER

Solver precondtion max steps
Type: integer **Default:** 1

SOLVER_PRECONDITION_STEPS

Solver precondition step type
Possible Values: CONSTANT, TRIANGULAR
Type: string **Default:** TRIANGULAR

SOLVER_QUANTITY

Solver residue or eigenvector delta
Possible Values: EIGENVECTOR, RESIDUAL
Type: string **Default:** RESIDUAL

SOLVER_TYPE

Solver type (for interchangeable solvers)
Possible Values: DL, RAYLEIGH
Type: string **Default:** DL

TDHF_MEM_SAFETY_FACTOR

Memory safety factor for allocating JK
Type: double **Default:** 0.75

B.14 DCFT

Performs Density Cumulant Functional Theory computations

ALGORITHM

The algorithm to use for lambda and orbital updates
Possible Values: TWOSTEP, SIMULTANEOUS
Type: string **Default:** SIMULTANEOUS

AO_BASIS

The algorithm to use for the $\langle VV||VV\rangle$ terms.
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

CACHELEVEL

How to cache quantities within the DPD library
Type: integer **Default:** 2

DAMPING_PERCENTAGE

The amount (percentage) of damping to apply to the initial SCF procedures 0 will result in a full update, 100 will completely stall the update. A value around 20 (which corresponds to 20% of the previous iteration's density being mixed into the current iteration) can help in cases where oscillatory convergence is observed.
Type: double **Default:** 0.0

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 6

DIIS_MIN_VECS

Minimum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 3

DIIS_START_CONVERGENCE

Value of RMS lambda and SCF errors below which DIIS starts
Type: double **Default:** 1e-3

IGNORE_TAU

Don't include the tau terms?
Type: boolean **Default:** false

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.
Type: double **Default:** 1e-14

LAMBDA_MAXITER

Maximum number of lambda iterations per macro-iteration
Type: integer **Default:** 50

LOCK_OCC

Do force the occupation to be that of the SCF starting point?
Type: boolean **Default:** true

MAXITER

Maximum number of iterations
Type: integer **Default:** 40

MO_RELAX

Do relax the orbitals?

Type: boolean **Default:** true

R_CONVERGENCE

Convergence criterion for residuals (RMS error) in density cummulant equations. See the note at the beginning of Section B.

Type: double **Default:** 1e-10

SCF_D_CONVERGENCE

Convergence criterion for the SCF density (RMS error). See the note at the beginning of Section B.

Type: double **Default:** 1e-8

SCF_MAXITER

Maximum number of SCF iterations per cycle

Type: integer **Default:** 50

TAU_SQUARED

Do compute the DCFT energy with the τ^2 correction to τ?

Type: boolean **Default:** false

TIKHONOW_OMEGA

The shift applied to the denominator

Type: double **Default:** 0.0

TPDM

Do compute the full two particle density matrix at the end of the computation, for properties?

Type: boolean **Default:** false

B.15 DETCI

ACTIVE

An array giving the number of active orbitals (occupied plus unoccupied) per irrep (shorthand to make MCSCF easier to specify than using RAS keywords)

Type: array **Default:** No Default

AVG_STATES

Array giving the root numbers of the states to average in a state-averaged procedure such as SA-CASSCF. Root numbering starts from 1.

Type: array **Default:** No Default

AVG_WEIGHTS

Array giving the weights for each state in a state-averaged procedure

Type: array **Default:** No Default

A_RAS3_MAX

maximum number of alpha electrons in RAS III

Type: integer **Default:** -1

B_RAS3_MAX

maximum number of beta electrons in RAS III

Type: integer **Default:** -1

CC

Do coupled-cluster computation?

Type: boolean **Default:** false

CC_A_RAS3_MAX

maximum number of alpha electrons in RAS III, for CC

Type: integer **Default:** -1

CC_B_RAS3_MAX

maximum number of beta electrons in RAS III, for CC

Type: integer **Default:** -1

CC_EX_LEVEL

The CC excitation level

Type: integer **Default:** 2

CC_RAS34_MAX

maximum number of electrons in RAS III + IV, for CC

Type: integer **Default:** -1

CC_RAS3_MAX

maximum number of electrons in RAS III, for CC

Type: integer **Default:** -1

CC_RAS4_MAX

maximum number of electrons in RAS IV, for CC

Type: integer **Default:** -1

CC_VAL_EX_LEVEL

The CC valence excitation level

Type: integer **Default:** 0

CC_VECS_READ

Do import a CC vector from disk?

Type: boolean **Default:** false

CC_VECS_WRITE

Do export a CC vector to disk?

Type: boolean **Default:** false

CIBLKS_PRINT

Do print a summary of the CI blocks?

Type: boolean **Default:** false

COLLAPSE_SIZE

Gives the number of vectors to retain when the Davidson subspace is collapsed (see MAX_NUM_VECS below). If greater than one, the collapsed subspace retains the best estimate of the CI vector for the previous n iterations. Defaults to 1.

Type: integer **Default:** 1

DETCI_FREEZE_CORE

Do freeze core orbitals?

Type: boolean **Default:** true

DIAG_METHOD

This specifies which method is to be used in diagonalizing the Hamiltonian. The valid options are: RSP, to form the entire H matrix and diagonalize using libciomr to obtain all eigenvalues (n.b. requires HUGE memory); OLSEN, to use Olsen's preconditioned inverse subspace method (1990); MITRUSHENKOV, to use a 2x2 Olsen/Davidson method; and DAVIDSON (or SEM) to use Liu's Simultaneous Expansion Method, which is identical to the Davidson method if only one root is to be found. There also exists a SEM debugging mode, SEMTEST. The SEM method is the most robust, but it also requires 2(N*M)+1 CI vectors on disk, where N is the maximum number of iterations and M is the number of roots.

Type: string **Default:** SEM

DIIS

Do use DIIS extrapolation to accelerate CC convergence?

Type: boolean **Default:** true

DIIS_FREQ

How often to do a DIIS extrapolation. 1 means do DIIS every iteration, 2 is every other iteration, etc.

Type: integer **Default:** 1

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 5

DIIS_MIN_VECS

Minimum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 2

DIIS_START_ITER

Iteration at which to start using DIIS

Type: integer **Default:** 1

DIPMOM

Do compute the dipole moment?

Type: boolean **Default:** false

EX_LEVEL

The CI excitation level

Type: integer **Default:** 2

E_CONVERGENCE

Convergence criterion for energy. See the note at the beginning of Section B.

Type: double **Default:** 1e-6

FCI

Do a full CI (FCI)? If TRUE, overrides the value of EX_LEVEL

Type: boolean **Default:** false

FOLLOW_ROOT

The root to write out the two-particle density matrix for (the one-particle density matrices are written for all roots). Useful for a state-specific CASSCF or CI optimization on an excited state.

Type: integer **Default:** 1

ICORE

Specifies how to handle buffering of CI vectors. A value of 0 makes the program perform I/O one RAS subblock at a time; 1 uses entire CI vectors at a time; and 2 uses one irrep block at a time. Values of 0 or 2 cause some inefficiency in the I/O (requiring multiple reads of the C vector when constructing H in the iterative subspace if DIAG_METHOD = SEM), but require less core memory.

Type: integer **Default:** 1

ISTOP

Do stop DETCI after string information is formed and before integrals are read?

Type: boolean **Default:** false

LSE

Do use least-squares extrapolation in iterative solution of CI vector?

Type: boolean **Default:** false

LSE_COLLAPSE

Number of iterations between least-squares extrapolations

Type: integer **Default:** 3

LSE_TOLERANCE

Minimum converged energy for least-squares extrapolation to be performed

Type: double **Default:** 3

MAXITER

Maximum number of iterations to diagonalize the Hamiltonian

Type: integer **Default:** 12

MAX_NUM_VECS

Gives the maximum number of Davidson subspace vectors which can be held on disk for the CI coefficient and sigma vectors. (There is one H(diag) vector and the number of D vectors is equal to the number of roots). When the number of vectors on disk reaches the value of MAX_NUM_VECS, the Davidson subspace will be collapsed to COLLAPSE_SIZE vectors for each root. This is very helpful for saving disk space. Defaults to MAXITER * NUM_ROOTS + NUM_INIT_VECS.

Type: integer **Default:** 0

MPN

Do compute the MPn series out to kth order where k is determined by MAX_NUM_VECS? For open-shell systems (REF=ROHF, WFN = ZAPTN), DETCI will compute the ZAPTn series. GUESS_VECTOR must be set to UNIT, HD_OTF must be set to TRUE, and HD_AVG must be set to orb_ener; these should happen by default for MPN=TRUE.

Type: boolean **Default:** false

MS0

Do use the $M_s = 0$ component of the state? Defaults to TRUE if closed-shell and FALSE otherwise. Related to the S option.

Type: boolean **Default:** false

NAT_ORBS_WRITE

Do write the natural orbitals?

Type: boolean **Default:** false

NAT_ORBS_WRITE_ROOT

Sets the root number for which CI natural orbitals are written to PSIF_CHKPT. The default value is 1 (lowest root).

Type: integer **Default:** 1

NUM_AMPS_PRINT

Number of important CC amplitudes per excitation level to print. CC analog to NUM_DETS_PRINT

Type: integer **Default:** 10

NUM_DETS_PRINT

Number of important determinants to print

Type: integer **Default:** 20

NUM_ROOTS

number of CI roots to find

Type: integer **Default:** 1

NUM_THREADS

Number of threads

Type: integer **Default:** 1

NUM_VECS_WRITE

Number of vectors to export

Type: integer **Default:** 1

OPDM

Do compute one-particle density matrix if not otherwise required?

Type: boolean **Default:** false

OPDM_AVG

Do average the OPDM over several roots in order to obtain a state-average one-particle density matrix? This density matrix can be diagonalized to obtain the CI natural orbitals.

Type: boolean **Default:** false

OPDM_PRINT

Do print the one-particle density matrix for each root?

Type: boolean **Default:** false

PRECONDITIONER

This specifies the type of preconditioner to use in the selected diagonalization method. The valid options are: DAVIDSON which approximates the Hamiltonian matrix by the diagonal elements; H0BLOCK_INV which uses an exact Hamiltonian of H0_BLOCKSIZE and explicitly inverts it; GEN_DAVIDSON which does a spectral decomposition of H0BLOCK; ITER_INV using an iterative approach to obtain the correction vector of H0BLOCK. The H0BLOCK_INV, GEN_DAVIDSON, and ITER_INV approaches are all formally equivalent but the ITER_INV is less computationally expensive. Default is DAVIDSON.

Type: string **Default:** DAVIDSON

RAS34_MAX

maximum number of electrons in RAS III + IV

Type: integer **Default:** -1

RAS3_MAX

maximum number of electrons in RAS III

Type: integer **Default:** -1

RAS4_MAX

maximum number of electrons in RAS IV

Type: integer **Default:** -1

REFERENCE

Reference wavefunction type

Possible Values: RHF, ROHF

Type: string **Default:** RHF

RESTART

Do result a DETCI iteration that terminated prematurely? It assumes that the CI and sigma vectors are on disk; the number of vectors specified by RESTART_VECS is collapsed down to one vector per root.

Type: boolean **Default:** false

RESTRICTED_DOCC

An array giving the number of restricted doubly-occupied orbitals per irrep (not excited in CI wavefunctions, but orbitals can be optimized in MCSCF)

Type: array **Default:** No Default

RESTRICTED_UOCC

An array giving the number of restricted unoccupied orbitals per irrep (not occupied in CI wavefunctions, but orbitals can be optimized in MCSCF)

Type: array **Default:** No Default

R_CONVERGENCE

Convergence criterion for CI residual vector in the Davidson algorithm (RMS error). The default is 1e-4 for energies and 1e-7 for gradients. See the note at the beginning of Section B.

Type: double **Default:** 1e-4

S

The value of the spin quantum number S is given by this option. The default is determined by the value of the multiplicity. This is used for two things: (1) determining the phase of the redundant half of the CI vector when the Ms=0 component is used (i.e., Ms0 = TRUE), and (2) making sure the guess vector has the desired value of $\langle S^2 \rangle$ (if S_SQUARED is TRUE and ICORE=1).

Type: double **Default:** 0.0

S_SQUARED

Do calculate the value of $\langle S^2 \rangle$ for each root?

Type: boolean **Default:** false

TDM

Do compute the transition density? Note: only transition densities between roots of the same symmetry will be evaluated. DETCI does not compute states of different irreps within the same computation; to do this, lower the symmetry of the computation.

Type: boolean **Default:** false

TDM_PRINT

Do print the transition density?

Type: boolean **Default:** false

TDM_WRITE

Do write the transition density?

Type: boolean **Default:** false

TPDM

Do compute two-particle density matrix if not otherwise required?

Type: boolean **Default:** false

TPDM_PRINT

Do print the two-particle density matrix? (Warning: large tensor)
Type: boolean **Default:** false

UPDATE

DAVIDSON employs the standard DAVIDSON update or correction vector formula, while OLSEN uses the OLSEN correction vector. Default is DAVIDSON.
Possible Values: DAVIDSON, OLSEN
Type: string **Default:** DAVIDSON

VAL_EX_LEVEL

In a RAS CI, this is the additional excitation level for allowing electrons out of RAS I into RAS II. The maximum number of holes in RAS I is therefore EX_LEVEL + VAL_EX_LEVEL.
Type: integer **Default:** 0

VECS_WRITE

Do store converged vector(s) at the end of the run? The vector(s) is(are) stored in a transparent format such that other programs can use it easily. The format is specified in src/lib/libqt/slaterdset.h.
Type: boolean **Default:** false

B.16 DFCC

Performs density-fitted coupled cluster computations.

BASIS

Primary basis set
Type: string **Default:** NONE

DEALIAS_BASIS_CC

Dealias basis for PS integrals
Type: string **Default:** No Default

DEALIAS_BASIS_SCF

Random stuff for LibFock
Type: string **Default:** No Default

DEALIAS_BETA

Dealias basis beta parameter
Type: double **Default:** 3.5

DEALIAS_DELTA

Dealias basis delta parameter
Type: double **Default:** 2.0

DEALIAS_N_CAP

Dealias basis N cap parameter
Type: integer **Default:** 1

DEALIAS_N_CORE

Dealias basis N core parameter
Type: integer **Default:** 1

DEALIAS_N_DIFFUSE

Dealias basis N diffuse parameter
Type: integer **Default:** 1

DEALIAS_N_INTERCALATER

Dealias basis N intercalater parameter
Type: integer **Default:** 1

DEALIAS_N_L

Dealias basis highest delta l parameter
Type: integer **Default:** 1

DENOMINATOR_ALGORITHM

Denominator algorithm for PT methods
Possible Values: LAPLACE, CHOLESKY
Type: string **Default:** LAPLACE

DENOMINATOR_DELTA

Maximum denominator error allowed (Max error norm in Delta tensor)
Type: double **Default:** 1.0e-6

DF_BASIS_CC

Auxiliary basis set for density fitting MO integrals. Defaults to BASIS-RI.
Type: string **Default:** NONE

DIIS

Do use DIIS extrapolation to accelerate convergence?
Type: boolean **Default:** true

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 6

DIIS_MIN_VECS

Minimum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 2

E_CONVERGENCE

Convergence criterion for CC energy. See the note at the beginning of Section B.
Type: double **Default:** 1e-8

FITTING_COND

Desired Fitting condition (inverse of max condition number)
Type: double **Default:** 1.0e-10

FITTING_TYPE

Fitting metric algorithm
Possible Values: EIG, CHOLESKY, QR
Type: string **Default:** EIG

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double **Default:** 0.0

MAXITER

Maximum number iterations

Type: integer **Default:** 40

MP2_ALGORITHM

MP2 Algorithm:

Algorithm Keyword	MP2J	MP2K
MP2	MP2	MP2
DF	DF	DF
PS	PS	PS
PS1	DF	PS/DF
PS2	DF	PS/PS
PS3	PS	PS/DF
PS4	PS	PS/PS
TEST_DENOM	Test	Test
TEST_PS	Test	Test
TEST_PS_OMEGA	Test	Test
TEST_DPS_OMEGA	Test	Test
TEST_DF	Test	Test

Possible Values: MP2, DF, PS, PS1, PS2, PS3, PS4, TEST_DENOM, TEST_PS, TEST_PS_OMEGA, TEST_DPS_OMEGA, TEST_DF

Type: string **Default:** DF

MP2_OS_SCALE

OS Scale

Type: double **Default:** 6.0/5.0

MP2_SS_SCALE

SS Scale

Type: double **Default:** 1.0/3.0

PS_ALPHA

Pseudospectral partition alpha

Type: double **Default:** 1.0

PS_BASIS_TOLERANCE

PS basis cutoff. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-12

PS_BLOCK_MAX_POINTS

The maximum number of grid points per evaluation block.

Type: integer **Default:** 5000

PS_BLOCK_MAX_RADIUS

The maximum radius to terminate subdivision of an octree block (a.u.)

Type: double **Default:** 1.0

PS_BLOCK_MIN_POINTS

The minimum number of grid points per evaluation block.

Type: integer **Default:** 1000

PS_BS_RADIUS_ALPHA

Factor for effective BS radius in radial grid

Type: double **Default:** 1.0

PS_DEALIASING

Random stuff for LibFock

Possible Values: QUADRATURE, RENORMALIZED, DEALIASED

Type: string **Default:** QUADRATURE

PS_FITTING_ALGORITHM

Fitting algorithm to use for pseudospectral

Possible Values: DEALIASED, RENORMALIZED, QUADRATURE

Type: string **Default:** CONDITIONED

PS_GRID_FILE

Filename to read grid from

Type: string **Default:** No Default

PS_GRID_PATH

File path to read grids from
Type: string **Default:** No Default

PS_MIN_S_DEALIAS

Minumum eigenvalue for dealias basis
Type: double **Default:** 1.0e-7

PS_MIN_S_PRIMARY

Minumum eigenvalue for primary basis
Type: double **Default:** 1.0e-7

PS_NUCLEAR_SCHEME

Nuclear Scheme
Possible Values: TREUTLER, BECKE, NAIVE, STRATMANN
Type: string **Default:** TREUTLER

PS_NUM_RADIAL

Number of radial points
Type: integer **Default:** 5

PS_OMEGA

Pseudospectral range-separation parameter
Type: double **Default:** 1.0

PS_ORDER_SPHERICAL

Maximum order of spherical grids
Type: integer **Default:** 7

PS_PRUNING_ALPHA

Spread alpha for logarithmic pruning
Type: double **Default:** 1.0

PS_PRUNING_SCHEME

Pruning Scheme
Possible Values: FLAT, P_GAUSSIAN, D_GAUSSIAN, P_SLATER, D_SLATER, LOG_GAUSSIAN, LOG_SLATER
Type: string **Default:** FLAT

PS_RADIAL_SCHEME

Radial Scheme
Possible Values: TREUTLER, BECKE, MULTIEXP, EM, MURA
Type: string **Default:** TREUTLER

PS_SPHERICAL_SCHEME

Spherical Scheme
Possible Values: LEBEDEV
Type: string **Default:** LEBEDEV

PS_THETA

Random stuff for LibFock
Type: double **Default:** 0.3

PS_USE_OMEGA

Do use range-separation procedure in PS?
Type: boolean **Default:** true

RPA_ALGORITHM

RPA algorithm: $\begin{array}{ll} \text{DF} & \mathcal{O}(N^5) \\ \text{CD} & \mathcal{O}(N^4) \end{array}$
Possible Values: CD, DF
Type: string **Default:** CD

RPA_ALPHA

RPA alpha parameter
Type: double **Default:** 1.0

RPA_DELTA

RPA Cholesky delta
Type: double **Default:** 1.0e-6

RPA_PLUS_EPSILON

Continue RPA numerical SPD Tolerance
Type: double **Default:** 1.0e-12

RPA_RISKY

Do continue RPA even if T's are not numerically SPD?
Type: boolean **Default:** false

R_CONVERGENCE

Convergence criterion for cluster amplitudes (RMS change). See the note at the beginning of Section B.
Type: double **Default:** 1e-8

WAVEFUNCTION

Type of wavefunction
Possible Values: MP2, MP3, CCD, DRPA
Type: string **Default:** MP2

B.17 DFMP2

Performs density-fitted MP2 computations for RHF/UHF/ROHF reference wavefunctions.

BASIS

Primary basis set
Type: string **Default:** NONE

DFMP2_MEM_FACTOR

% of memory for DF-MP2 three-index buffers
Type: double **Default:** 0.9

DF_BASIS_MP2

Auxiliary basis set for MP2 density fitting computations. Defaults to BASIS-RI.
Type: string **Default:** No Default

DF_INTS_NUM_THREADS

Number of threads to compute integrals with. 0 is wild card

Type: integer	**Default:** 0

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double	**Default:** 0.0

MP2_OS_SCALE

OS Scale

Type: double	**Default:** 6.0/5.0

MP2_SS_SCALE

SS Scale

Type: double	**Default:** 1.0/3.0

B.18 FINDIF

Performs finite difference computations of energy derivative, with respect to nuclear displacements for geometry optimizations and vibrational frequency analyses, where the required analytical derivatives are not available.

DISP_SIZE

Displacement size in au for finite-differences.

Type: double	**Default:** 0.005

POINTS

Number of points for finite-differences (3 or 5)

Type: integer	**Default:** 3

B.19 LMP2

Performs local MP2 computations for RHF reference functions

DF_BASIS_MP2

Auxiliary basis set for MP2 density fitting calculations

Type: string **Default:** No Default

DF_LMP2

Do use density fitting? Turned on with specification of fitting basis.

Type: boolean **Default:** true

DIIS

Do use DIIS extrapolation to accelerate convergence?

Type: boolean **Default:** true

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 5

DIIS_START_ITER

Iteration at which to start DIIS extrapolation

Type: integer **Default:** 3

DISTANT_PAIR_CUTOFF

Distant pair cutoff

Type: double **Default:** 8.0

DOMAIN_PRINT_EXIT

Do exit after printing the domains?

Type: boolean **Default:** false

E_CONVERGENCE

Convergence criterion for energy (change). See the note at the beginning of Section B.

Type: double **Default:** 1e-7

FOCK_TOLERANCE

Minimum absolute value below which parts of the Fock matrix are skipped. See the note at the beginning of Section B.

Type: double **Default:** 1e-2

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double **Default:** 1e-7

LOCAL_CUTOFF

Localization cutoff

Type: double **Default:** 0.02

MAXITER

Maximum number of iterations

Type: integer **Default:** 50

MEMORY

Type: integer **Default:** 2000

MP2_OS_SCALE

Type: double **Default:** 6.0/5.0

MP2_SS_SCALE

Type: double **Default:** 1.0/3.0

NEGLECT_DISTANT_PAIR

Do neglect distant pairs?

Type: boolean **Default:** true

REFERENCE

Reference wavefunction type
Possible Values: RHF
Type: string **Default:** RHF

R_CONVERGENCE

Convergence criterion for T2 amplitudes (RMS change). See the note at the beginning of Section B.
Type: double **Default:** 1e-5

SCREEN_INTS

Do screen integrals?
Type: boolean **Default:** false

SCS

Do ?
Type: boolean **Default:** false

SCS_N

Do ?
Type: boolean **Default:** false

B.20 MCSCF

Performs RHF/UHF/ROHF/TCSCF, and more general MCSCF computations. Called as the starting point for multireference coupled cluster computations.

CANONICALIZE_ACTIVE_FAVG

Do canonicalize the active orbitals such that the average Fock matrix is diagonal?
Type: boolean **Default:** false

CANONICALIZE_INACTIVE_FAVG

Do canonicalize the inactive (DOCC and Virtual) orbitals such that the average Fock matrix is diagonal?
Type: boolean **Default:** false

CI_DIIS

Do use DIIS extrapolation to accelerate convergence of the CI coefficients?

Type: boolean **Default:** false

DIIS

Do use DIIS extrapolation to accelerate convergence of the SCF energy (MO coefficients only)?

Type: boolean **Default:** true

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 7

DOCC

The number of doubly occupied orbitals, per irrep

Type: array **Default:** No Default

D_CONVERGENCE

Convergence criterion for density. See the note at the beginning of Section B.

Type: double **Default:** 1e-12

E_CONVERGENCE

Convergence criterion for energy. See the note at the beginning of Section B.

Type: double **Default:** 1e-12

FAVG

Do use the average Fock matrix during the SCF optimization?

Type: boolean **Default:** false

FAVG_START

Iteration at which to begin using the averaged Fock matrix

Type: integer **Default:** 5

FOLLOW_ROOT

Which solution of the SCF equations to find, where 1 is the SCF ground state

Type: integer **Default:** 1

FORCE_TWOCON

Do attempt to force a two configruation solution by starting with CI coefficents of $\pm\sqrt{\frac{1}{2}}$

Type: boolean **Default:** false

INTERNAL_ROTATIONS

Do ?

Type: boolean **Default:** true

LEVEL_SHIFT

Level shift to aid convergence

Type: double **Default:** 0.0

MAXITER

Maximum number of iterations

Type: integer **Default:** 100

MO_READ

Do read in from file the MOs from a previous computation?

Type: boolean **Default:** true

REFERENCE

Reference wavefunction type

Possible Values: RHF, ROHF, UHF, TWOCON, MCSCF, GENERAL

Type: string **Default:** RHF

SOCC

The number of singly occupied orbitals, per irrep

Type: array **Default:** No Default

TURN_ON_ACTV

Type: integer **Default:** 0

WFN_SYM

The symmetry of the SCF wavefunction.
Possible Values: A, AG, AU, AP, APP, A1, A2, B, BG, BU, B1, B2, B3, B1G, B2G, B3G, B1U, B2U, B3U, 0, 1, 2, 3, 4, 5, 6, 7, 8
Type: string **Default:** 1

B.21 MINTS

Called at the beginning of SCF computations, whenever disk-based molecular integrals are required

BASIS

Primary basis set
Type: string **Default:** No Default

B.22 MP2

Performs second order Moller-Plesset perturbation theory (MP2) computations. This code can compute RHF/ROHF/UHF energies, and RHF gradient/property computations. However, given the small errors introduced, we recommend using the new density fitted MP2 codes instead, which are much more efficient.

CACHELEVEL

The amount of cacheing of data to perform
Type: integer **Default:** 2

CACHETYPE

The criterion used to retain/release cached data
Possible Values: LRU, LOW
Type: string **Default:** LRU

MP2_OS_SCALE

The scale factor used for opposite-spin pairs in SCS computations
Type: double **Default:** 6.0/5.0

MP2_SS_SCALE

The scale factor used for same-spin pairs in SCS computations
Type: double **Default:** 1.0/3.0

OPDM

Do compute the one particle density matrix, for properties?

Type: boolean **Default:** false

OPDM_RELAX

Do add relaxation terms to the one particle density matrix, for properties?

Type: boolean **Default:** false

REFERENCE

Reference wavefunction type

Possible Values: RHF, UHF, ROHF

Type: string **Default:** RHF

SCS

Do perform a spin component scaled MP2 computation?

Type: boolean **Default:** false

SCS_N

Do perform a spin component scaled (N) MP2 computation?

Type: boolean **Default:** false

B.23 MRCC

Interface to MRCC program written by Mihály Kállay.

E_CONVERGENCE

See the note at the beginning of Section B. This becomes `tol` (option #16) in fort.56.

Type: double **Default:** 1e-8

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-12

MRCC_LEVEL

Maximum excitation level. This is used ONLY if it is explicity set by the user. Single-reference case: all excitations up to this level are included, e.g., 2 for CCSD, 3 for CCSDT, 4 for CCSDTQ, etc. This becomes `ex.lev` (option #1) in fort.56.

Type: integer **Default:** 2

MRCC_NUM_SINGLET_ROOTS

Number of singlet roots. (Strictly speaking number of of roots with M_s=0 and S is even.) Use this option only with closed shell reference determinant, it must be zero otherwise. This becomes `nsing` (option #2) in fort.56.

Type: integer **Default:** 1

MRCC_NUM_TRIPLET_ROOTS

Number of triplet roots. (Strictly speaking number of of roots with M_s=0 and S is odd.) See notes at option MRCC_NUM_SINGLET_ROOTS. This becomes `ntrip` (option #3) in fort.56.

Type: integer **Default:** 0

B.24 OMP2

Performs quadratically convergence orbital-optimized MP2 computations.

CACHELEVEL

Type: integer **Default:** 2

CC_MAXITER

Type: integer **Default:** 50

CUTOFF

Type: integer **Default:** 14

DIIS_MAX_VECS

Number of vectors used in DIIS

Type: integer **Default:** 4

DO_SCS

Do ?

Type: boolean **Default:** false

DO_SOS

Do ?

Type: boolean **Default:** false

E_CONVERGENCE

See the note at the beginning of Section B.

Type: double **Default:** 1e-8

HESS_TYPE

Possible Values: NONE

Type: string **Default:** NONE

LEVEL_SHIFT

Type: double **Default:** 0.02

MAX_MOGRAD_CONVERGENCE

See the note at the beginning of Section B.

Type: double **Default:** 1e-4

MO_MAXITER

Type: integer **Default:** 50

MO_READ

Do read coefficient matrices from psi files?

Type: boolean **Default:** false

MO_STEP_MAX

Type: double **Default:** 0.5

MO_WRITE

Do write coefficient matrices to psi files?

Type: boolean **Default:** false

MP2_OS_SCALE

Type: double **Default:** 6.0/5.0

MP2_SS_SCALE

Type: double **Default:** 1.0/3.0

NAT_ORBS

Do ?

Type: boolean **Default:** false

OMP2_ORBS_PRINT

Do ?

Type: boolean **Default:** false

OPT_METHOD

Possible Values: SD, DIIS

Type: string **Default:** DIIS

ORTH_TYPE

Possible Values: GS, MGS

Type: string **Default:** MGS

RMS_MOGRAD_CONVERGENCE

See the note at the beginning of Section B.

Type: double **Default:** 1e-5

R_CONVERGENCE

See the note at the beginning of Section B.

Type: double **Default:** 1e-5

SOS_SCALE

Type: double **Default:** 1.3

SOS_SCALE2

Type: double **Default:** 1.2

B.25 OPTKING

Performs geometry optimizations and vibrational frequency analyses.

ADD_AUXILIARY_BONDS

Do add bond coordinates at nearby atoms for non-bonded systems?

Type: boolean **Default:** false

CART_HESS_READ

Do read Cartesian Hessian? Only for experts - use FULL_HESS_EVERY instead.

Type: boolean **Default:** false

CONSECUTIVE_BACKSTEPS

Set number of consecutive backward steps allowed in optimization

Type: integer **Default:** 0

COVALENT_CONNECT

When determining connectivity, a bond is assigned if interatomic distance is less than (this number) * sum of covalent radii double

Type: double **Default:** 1.3

FINAL_GEOM_WRITE

Do save and print the geometry from the last projected step at the end of a geometry optimization? Otherwise (and by default), save and print the previous geometry at which was computed the gradient that satisfied the convergence criteria.

Type: boolean **Default:** false

FLEXIBLE_G_CONVERGENCE

Even if a user-defined threshold is set, allow for normal, flexible convergence criteria

Type: boolean **Default:** false

FRAG_MODE

For multi-fragment molecules, treat as single bonded molecule or via interfragment coordinates. A primary difference is that in MULTI mode, the interfragment coordinates are not redundant.
Possible Values: SINGLE, MULTI

Type: string **Default:** SINGLE

FREEZE_INTRAFRAG

Do freeze all fragments rigid?

Type: boolean **Default:** false

G_CONVERGENCE

Set of optimization criteria. Specification of MAX_ or RMS_ G_CONVERGENCE options will append or overwrite the criteria set here.
Possible Values: QCHEM, MOLPRO, GAU, GAU_LOOSE, GAU_TIGHT, GAU_VERYTIGHT, TURBOMOLE, CFOUR, NWCHEM_LOOSE

Type: string **Default:** QCHEM

HESS_UPDATE

Hessian update scheme
Possible Values: NONE, BFGS, MS, POWELL, BOFILL

Type: string **Default:** BFGS

HESS_UPDATE_LIMIT

Do limit the magnitude of changes caused by the Hessian update?

Type: boolean **Default:** true

HESS_UPDATE_LIMIT_MAX

If HESS_UPDATE_LIMIT is true, changes to the Hessian from the update are limited to the larger of (HESS_UPDATE_LIMIT_SCALE)*(the previous value) and HESS_UPDATE_LIMIT_MAX (in au).

Type: double **Default:** 1.00

HESS_UPDATE_LIMIT_SCALE

If the above is true, changes to the Hessian from the update are limited to the larger of (HESS_UPDATE_LIMIT_SCALE)*(the previous value) and HESS_UPDATE_LIMIT_MAX (in au).

Type: double **Default:** 0.50

HESS_UPDATE_USE_LAST

Number of previous steps to use in Hessian update, 0 uses all

Type: integer **Default:** 1

H_BOND_CONNECT

For now, this is a general maximum distance for the definition of H-bonds

Type: double **Default:** 4.3

INTCOS_GENERATE_EXIT

Do only generate the internal coordinates and then stop?

Type: boolean **Default:** false

INTERFRAG_DIST_INV

Do use $\frac{1}{R_{AB}}$ for the stretching coordinate between fragments? Otherwise, use R_{AB}.

Type: boolean **Default:** false

INTERFRAG_HESS

Whether to use the default of FISCHER_LIKE force constants for the initial guess DEFAULT, FISCHER_LIKE

Possible Values: DEFAULT, FISCHER_LIKE

Type: string **Default:** DEFAULT

INTERFRAG_MODE

When interfragment coordinates are present, use as reference points either principal axes or fixed linear combinations of atoms.
Possible Values: FIXED, INTERFRAGMENT
Type: string **Default:** FIXED

INTRAFRAG_HESS

What model Hessian to use to guess intrafragment force constants SCHLEGEL, FISCHER, SIMPLE
Possible Values: FISCHER, SCHLEGEL, SIMPLE
Type: string **Default:** SCHLEGEL

INTRAFRAG_STEP_LIMIT

Initial maximum step size in bohr or radian along an internal coordinate
Type: double **Default:** 0.4

INTRAFRAG_STEP_LIMIT_MAX

Upper bound for dynamic trust radius [au]
Type: double **Default:** 1.0

INTRAFRAG_STEP_LIMIT_MIN

Lower bound for dynamic trust radius [au]
Type: double **Default:** 0.001

IRC_DIRECTION

IRC mapping direction
Possible Values: FORWARD, BACKWARD
Type: string **Default:** FORWARD

IRC_STEP_SIZE

IRC step size in bohr(amu)$^{1/2}$
Type: double **Default:** 0.2

MAX_DISP_G_CONVERGENCE

Convergence criterion for geometry optmization: maximum displacement (internal coordinates, atomic units). See the note at the beginning of Section B.
Type: double **Default:** 1.2e-3

MAX_ENERGY_G_CONVERGENCE

Convergence criterion for geometry optmization: maximum energy change. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-6

MAX_FORCE_G_CONVERGENCE

Convergence criterion for geometry optmization: maximum force (internal coordinates, atomic units). See the note at the beginning of Section B.

Type: double **Default:** 3.0e-4

OPT_TYPE

Specifies minimum search, transition-state search, or IRC following
Possible Values: MIN, TS, IRC

Type: string **Default:** MIN

RFO_FOLLOW_ROOT

Do follow the initial RFO vector after the first step?

Type: boolean **Default:** false

RFO_ROOT

Root for RFO to follow, 0 being lowest (for a minimum)

Type: integer **Default:** 0

RMS_DISP_G_CONVERGENCE

Convergence criterion for geometry optmization: rms displacement (internal coordinates, atomic units). See the note at the beginning of Section B.

Type: double **Default:** 1.2e-3

RMS_FORCE_G_CONVERGENCE

Convergence criterion for geometry optmization: rms force (internal coordinates, atomic units). See the note at the beginning of Section B.

Type: double **Default:** 3.0e-4

STEP_TYPE

Geometry optimization step type, either Newton-Raphson or Rational Function Optimization
Possible Values: RFO, NR, SD

Type: string **Default:** RFO

TEST_B

Do test B matrix?
Type: boolean **Default:** false

TEST_DERIVATIVE_B

Do test derivative B matrix?
Type: boolean **Default:** false

B.26 PLUGIN-CCSD-SERIAL

B.27 PSIMRCC

Performs multireference coupled cluster computations. This theory should be used only by advanced users with a good working knowledge of multireference techniques.

ACTIVE

The number of active orbitals per irrep
Type: array **Default:** No Default

CORR_ANSATZ

The ansatz to use for MRCC computations
Possible Values: SR, MK, BW, APBW
Type: string **Default:** MK

CORR_CCSD_T

The type of CCSD(T) computation to perform
Possible Values: STANDARD, PITTNER
Type: string **Default:** STANDARD

CORR_CHARGE

The molecular charge of the target state
Type: integer **Default:** 0

CORR_MULTP

The multiplicity, $M_S(M_S+1)$, of the target state. Must be specified if different from the reference M_s.
Type: integer **Default:** 1

CORR_REFERENCE

Reference wavefunction type used in MRCC computations
Possible Values: RHF, ROHF, TCSCF, MCSCF, GENERAL
Type: string **Default:** GENERAL

CORR_WFN

The type of correlated wavefunction
Possible Values: PT2, CCSD, MP2-CCSD, CCSD_T
Type: string **Default:** CCSD

COUPLING

The order of coupling terms to include in MRCCSDT computations
Possible Values: NONE, LINEAR, QUADRATIC, CUBIC
Type: string **Default:** CUBIC

COUPLING_TERMS

Do include the terms that couple the reference determinants?
Type: boolean **Default:** true

DAMPING_PERCENTAGE

The amount (percentage) of damping to apply to the amplitude updates. 0 will result in a full update, 100 will completely stall the update. A value around 20 (which corresponds to 20% of the amplitudes from the previous iteration being mixed into the current iteration) can help in cases where oscillatory convergence is observed.
Type: double **Default:** 0.0

DIAGONALIZE_HEFF

Do diagonalize the effective Hamiltonian?
Type: boolean **Default:** false

DIAGONAL_CCSD_T

Do include the diagonal corrections in (T) computations?
Type: boolean **Default:** true

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 7

DIIS_START

The number of DIIS vectors needed before extrapolation is performed

Type: integer **Default:** 2

E_CONVERGENCE

Convergence criterion for energy. See the note at the beginning of Section B.

Type: double **Default:** 1e-9

FAVG_CCSD_T

Do use the averaged Fock matrix over all references in (T) computations?

Type: boolean **Default:** false

FOLLOW_ROOT

Which root of the effective hamiltonian is the target state?

Type: integer **Default:** 1

FROZEN_DOCC

The number of frozen occupied orbitals per irrep

Type: array **Default:** No Default

FROZEN_UOCC

The number of frozen virtual orbitals per irrep

Type: array **Default:** No Default

HEFF4

Do include the fourth-order contributions to the effective Hamiltonian?

Type: boolean **Default:** true

HEFF_PRINT

Do print the effective Hamiltonian?

Type: boolean **Default:** false

LOCK_SINGLET

Do lock onto a singlet root?

Type: boolean **Default:** false

MAXITER

Maximum number of iterations to determine the amplitudes
Type: integer **Default:** 100

MP2_CCSD_METHOD

How to perform MP2_CCSD computations
Possible Values: I, IA, II
Type: string **Default:** II

MP2_GUESS

Do start from a MP2 guess?
Type: boolean **Default:** true

NO_SINGLES

Do ?
Type: boolean **Default:** false

NUM_THREADS

Number of threads
Type: integer **Default:** 1

OFFDIAGONAL_CCSD_T

Do include the off-diagonal corrections in (T) computations?
Type: boolean **Default:** true

PT_ENERGY

The type of perturbation theory computation to perform
Possible Values: SECOND_ORDER, SCS_SECOND_ORDER, PSEUDO_SECOND_ORDER, SCS_PSEUDO_SECOND_ORDER
Type: string **Default:** SECOND_ORDER

RESTRICTED_DOCC

The number of doubly occupied orbitals per irrep
Type: array **Default:** No Default

R_CONVERGENCE

Convergence criterion for amplitudes (residuals). See the note at the beginning of Section B.

Type: double **Default:** 1e-9

SMALL_CUTOFF

Type: integer **Default:** 0

TIKHONOW_MAX

The cycle after which Tikhonow regularization is stopped. Set to zero to allow regularization in all iterations

Type: integer **Default:** 5

TIKHONOW_OMEGA

The shift to apply to the denominators, *c.f.* Taube and Bartlett, JCP, 130, 144112 (2009)

Type: double **Default:** 0.0

TRIPLES_ALGORITHM

The type of algorithm to use for (T) computations
Possible Values: SPIN_ADAPTED, RESTRICTED, UNRESTRICTED

Type: string **Default:** RESTRICTED

TRIPLES_DIIS

Do use DIIS extrapolation to accelerate convergence for iterative triples excitations?

Type: boolean **Default:** false

USE_SPIN_SYM

Do use symmetry to map equivalent determinants onto each other, for efficiency?

Type: boolean **Default:** true

WFN_SYM

The symmetry of the target wavefunction, specified either by Schönflies symbol, or irrep number (in Cotton ordering)
Possible Values: A, AG, AU, AP, APP, A1, A2, B, BG, BU, B1, B2, B3, B1G, B2G, B3G, B1U, B2U, B3U, 0, 1, 2, 3, 4, 5, 6, 7, 8

Type: string **Default:** 1

ZERO_INTERNAL_AMPS

Do zero the internal amplitudes, i.e., those that map reference determinants onto each other?
Type: boolean **Default:** true

B.28 RESPONSE

Performs SCF linear response computations.

OMEGA

Type: array **Default:** No Default

PROPERTY

Array that specifies the desired frequencies of the incident radiation field in CCLR calculations. If only one element is given, the units will be assumed to be atomic units. If more than one element is given, then the units must be specified as the final element of the array. Acceptable units are HZ, NM, EV, and AU.
Possible Values: POLARIZABILITY, ROTATION, ROA, ALL
Type: string **Default:** POLARIZABILITY

REFERENCE

Reference wavefunction type
Type: string **Default:** RHF

B.29 SAPT

Performs symmetry adapted perturbation theory (SAPT) analysis to quantitatively analyze noncovalent interactions.

AIO_CPHF

Do use asynchronous disk I/O in the solution of the CPHF equations? Use may speed up the computation slightly at the cost of spawning an additional thread.
Type: boolean **Default:** false

AIO_DF_INTS

Do use asynchronous disk I/O in the formation of the DF integrals? Use may speed up the computation slightly at the cost of spawning an additional thread.
Type: boolean **Default:** false

BASIS

Primary basis set, describes the monomer molecular orbitals

Type: string **Default:** No Default

DENOMINATOR_ALGORITHM

Denominator algorithm for PT methods. Laplace transformations are slightly more efficient.
Possible Values: LAPLACE, CHOLESKY

Type: string **Default:** LAPLACE

DENOMINATOR_DELTA

Maximum error allowed (Max error norm in Delta tensor) in the approximate energy denominators employed for most of the $E^{(20)}_{disp}$ and $E^{(20)}_{exch-disp}$ evaluation.

Type: double **Default:** 1.0e-6

DF_BASIS_ELST

Auxiliary basis set for SAPT Elst10 and Exch10 density fitting computations, may be important if heavier elements are involved. Defaults to BASIS-RI.

Type: string **Default:** No Default

DF_BASIS_SAPT

Auxiliary basis set for SAPT density fitting computations. Defaults to BASIS-RI.

Type: string **Default:** No Default

D_CONVERGENCE

Convergence criterion for residual of the CPHF coefficients in the SAPT $E^{(20)}_{ind,resp}$ term. See the note at the beginning of Section B.

Type: double **Default:** 1e-8

E_CONVERGENCE

Convergence criterion for energy (change) in the SAPT $E^{(20)}_{ind,resp}$ term during solution of the CPHF equations. See the note at the beginning of Section B.

Type: double **Default:** 1e-10

FREEZE_CORE

The scope of core orbitals to freeze in evaluation of SAPT $E^{(20)}_{disp}$ and $E^{(20)}_{exch-disp}$ terms. Recommended true for all SAPT computations
Possible Values: FALSE, TRUE, SMALL, LARGE

Type: string **Default:** FALSE

INTS_TOLERANCE

Minimum absolute value below which all three-index DF integrals and those contributing to four-index integrals are neglected. The default is conservative, but there isn't much to be gained from loosening it, especially for higher-order SAPT. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-12

MAXITER

Maxmum number of CPHF iterations

Type: integer **Default:** 50

NAT_ORBS

Do natural orbitals to speed up evaluation of the triples contribution to dispersion by truncating the virtual orbital space? Recommended true for all SAPT computations.

Type: boolean **Default:** false

NAT_ORBS_T2

Do use MP2 natural orbital approximations for the v^4 block of two-electron integrals in the evaluation of second-order T2 amplitudes? This approximation is promising for accuracy and computational savings, but it has not been rigorously tested.

Type: boolean **Default:** false

NO_RESPONSE

Don't solve the CPHF equations? Evaluate $E_{ind}^{(20)}$ and $E_{exch-ind}^{(20)}$ instead of their response-including coupterparts. Only turn on this option if the induction energy is not going to be used.

Type: boolean **Default:** false

OCC_TOLERANCE

Minimum occupation (eigenvalues of the MP2 OPDM) below which virtual natural orbitals are discarded for evaluating the triples contribution to dispersion. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-6

PRINT

The amount of information to print to the output file for the sapt module. For 0, only the header and final results are printed. For 1, (recommended for large calculations) some intermediate quantities are also printed.

Type: integer **Default:** 1

SAPT_LEVEL

The level of theory for SAPT
Possible Values: SAPT0, SAPT2, SAPT2+, SAPT2+3
Type: string **Default:** SAPT0

SAPT_MEM_CHECK

Do force SAPT2 and higher to die if it thinks there isn't enough memory? Turning this off is ill-advised.
Type: boolean **Default:** true

SAPT_MEM_SAFETY

Memory safety
Type: double **Default:** 0.9

SAPT_OS_SCALE

The scale factor used for opposite-spin pairs in SCS computations. SS/OS decomposition performed for $E_{disp}^{(20)}$ and $E_{exch-disp}^{(20)}$ terms.
Type: double **Default:** 6.0/5.0

SAPT_SS_SCALE

The scale factor used for same-spin pairs in SCS computations. SS/OS decomposition performed for $E_{disp}^{(20)}$ and $E_{exch-disp}^{(20)}$ terms.
Type: double **Default:** 1.0/3.0

B.30 SCF

Performs self consistent field (Hartree-Fock and Density Functional Theory) computations. These are the starting points for most computations, so this code is called in most cases.

B.30.1 Convergence Control/Stabilization

DAMPING_CONVERGENCE

The density convergence threshold after which damping is no longer performed, if it is enabled. It is recommended to leave damping on until convergence, which is the default. See the note at the beginning of Section B.
Type: double **Default:** 1.0e-18

DAMPING_PERCENTAGE

The amount (percentage) of damping to apply to the early density updates. 0 will result in a full update, 100 will completely stall the update. A value around 20 (which corresponds to 20% of the previous iteration's density being mixed into the current density) could help to solve problems with oscillatory convergence.

Type: double **Default:** 100.0

DIIS

Do use DIIS extrapolation to accelerate convergence?

Type: boolean **Default:** true

DIIS_MAX_VECS

Maximum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 10

DIIS_MIN_VECS

Minimum number of error vectors stored for DIIS extrapolation

Type: integer **Default:** 2

DIIS_START

The minimum iteration to start storing DIIS vectors

Type: integer **Default:** 1

D_CONVERGENCE

Convergence criterion for SCF density. See the note at the beginning of Section B.

Type: double **Default:** 1e-8

E_CONVERGENCE

Convergence criterion for SCF energy. See the note at the beginning of Section B.

Type: double **Default:** 1e-8

MAXITER

Maximum number of iterations

Type: integer **Default:** 100

MOM_OCC

The absolute indices of orbitals to excite from in MOM (+/- for alpha/beta)

Type: array **Default:** No Default

MOM_START

The iteration to start MOM on (or 0 for no MOM)

Type: integer **Default:** 0

MOM_VIR

The absolute indices of orbitals to excite to in MOM (+/- for alpha/beta)

Type: array **Default:** No Default

B.30.2 DFSCF Algorithm

DF_INTS_NUM_THREADS

Number of threads for integrals (may be turned down if memory is an issue). 0 is blank

Type: integer **Default:** 0

B.30.3 DFT

DFT_BASIS_TOLERANCE

DFT basis cutoff. See the note at the beginning of Section B.

Type: double **Default:** 1.0e-12

DFT_BLOCK_MAX_POINTS

The maximum number of grid points per evaluation block.

Type: integer **Default:** 5000

DFT_BLOCK_MAX_RADIUS

The maximum radius to terminate subdivision of an octree block (a.u.)

Type: double **Default:** 3.0

DFT_BLOCK_MIN_POINTS

The minimum number of grid points per evaluation block.

Type: integer **Default:** 1000

DFT_BLOCK_SCHEME

The blocking scheme for DFT.
Possible Values: NAIVE, OCTREE
Type: string **Default:** OCTREE

DFT_BS_RADIUS_ALPHA

Factor for effective BS radius in radial grid.
Type: double **Default:** 1.0

DFT_FUNCTIONAL

The DFT combined functional name (for now).
Type: string **Default:** No Default

DFT_GRID_NAME

The DFT grid specification, such as SG1.
Possible Values: SG1
Type: string **Default:** No Default

DFT_NUCLEAR_SCHEME

Nuclear Scheme.
Possible Values: TREUTLER, BECKE, NAIVE, STRATMANN
Type: string **Default:** TREUTLER

DFT_NUM_RADIAL

Number of radial points.
Type: integer **Default:** 99

DFT_OMEGA

The DFT Range-separation parameter
Type: double **Default:** 0.0

DFT_ORDER_SPHERICAL

Maximum order of spherical grids.
Type: integer **Default:** 29

DFT_PRUNING_ALPHA

Spread alpha for logarithmic pruning.
Type: double **Default:** 1.0

DFT_PRUNING_SCHEME

Pruning Scheme.
Possible Values: FLAT, P_GAUSSIAN, D_GAUSSIAN, P_SLATER, D_SLATER, LOG_GAUSSIAN, LOG_SLATER
Type: string **Default:** FLAT

DFT_RADIAL_SCHEME

Radial Scheme.
Possible Values: TREUTLER, BECKE, MULTIEXP, EM, MURA
Type: string **Default:** TREUTLER

DFT_SPHERICAL_SCHEME

Spherical Scheme.
Possible Values: LEBEDEV
Type: string **Default:** LEBEDEV

B.30.4 Environmental Effects

EXTERN

An ExternalPotential (built by Python or NULL/None)
Type: python **Default:** No Default

PERTURB_H

Perturb the Hamiltonian?
Type: boolean **Default:** false

PERTURB_MAGNITUDE

Size of the perturbation
Type: double **Default:** 0.0

PERTURB_WITH

The operator used to perturb the Hamiltonian, if requested
Possible Values: DIPOLE_X, DIPOLE_Y, DIPOLE_Z
Type: string **Default:** DIPOLE_X

PROCESS_GRID

SUBESCTION Parallel Runtime
Type: array **Default:** No Default

B.30.5 Fractional Occupation UHF/UKS

FRAC_DIIS

Do use DIIS extrapolation to accelerate convergence in frac?
Type: boolean **Default:** true

FRAC_LOAD

Do recompute guess from stored orbitals?
Type: boolean **Default:** false

FRAC_OCC

The absolute indices of occupied orbitals to fractionally occupy (+/- for alpha/beta)
Type: array **Default:** No Default

FRAC_RENORMALIZE

Do renormalize C matrices prior to writing to checkpoint?
Type: boolean **Default:** true

FRAC_START

The iteration to start fractionally occupying orbitals (or 0 for no fractional occupation)
Type: integer **Default:** 0

FRAC_VAL

The occupations of the orbital indices specified above ($0.0 \geq occ \geq 1.0$)
Type: array **Default:** No Default

B.30.6 General Wavefunction Info

BASIS

Primary basis set
Type: string **Default:** No Default

DF_BASIS_SCF

Auxiliary basis set for SCF density fitting computations. Defaults to BASIS-JKFIT.
Type: string **Default:** No Default

GUESS

The type of guess orbitals
Possible Values: CORE, GWH, SAD, READ
Type: string **Default:** CORE

INTS_TOLERANCE

Minimum absolute value below which TEI are neglected. See the note at the beginning of Section B.
Type: double **Default:** 0.0

REFERENCE

Reference wavefunction type
Possible Values: RHF, ROHF, UHF, CUHF, RKS, UKS
Type: string **Default:** RHF

SAVE_JK

Keep JK object for later use?
Type: boolean **Default:** false

SCF_TYPE

What algorithm to use for the SCF computation
Possible Values: DIRECT, DF, PK, OUT_OF_CORE, PS
Type: string **Default:** PK

S_ORTHOGONALIZATION

SO orthogonalization: symmetric or canonical?
Possible Values: SYMMETRIC, CANONICAL
Type: string **Default:** SYMMETRIC

S_TOLERANCE

Minimum S matrix eigenvalue to be used before compensating for linear dependencies. See the note at the beginning of Section B.
Type: double **Default:** 1e-7

B.31 STABLE

Performs wavefunction stability analysis. Called when specifically requested by the user

CACHELEVEL

Type: integer **Default:** 2

FOLLOW

Do follow the most negative eigenvalue of the Hessian towards a lower energy HF solution? Follow a UHF$\rightarrow$UHF instability of same symmetry?

Type: boolean **Default:** false

NUM_VECS_PRINT

Number of lowest MO Hessian eigenvalues to print

Type: integer **Default:** 0

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

ROTATION_SCHEME

Method for following eigenvectors, either 0 by angles or 1 by antisymmetric matrix.

Type: integer **Default:** 0

SCALE

Scale factor (between 0 and 1) for orbital rotation step

Type: double **Default:** 0.5

B.32 TRANSQT

The predecessor to Transqt2. Currently used by the configuration interaction codes, but it is being phased out.

AA_M_FILE

Type: integer **Default:** PSIF_MO_AA_TEI

AB_M_FILE

Type: integer **Default:** PSIF_MO_AB_TEI

AO_BASIS

The algorithm to use for the $\langle VV||VV\rangle$ terms
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

BB_M_FILE

Type: integer **Default:** PSIF_MO_BB_TEI

CHECK_C_ORTHONORM

Do ?
Type: boolean **Default:** false

DELETE_AO

Don't ?
Type: boolean **Default:** true

DELETE_RESTR_DOCC

Don't ?
Type: boolean **Default:** true

DELETE_TPDM

Don't ?
Type: boolean **Default:** true

DO_ALL_TEI

Do ?

Type: boolean **Default:** false

FIRST_TMP_FILE

Type: integer **Default:** 150

FZC_A_FILE

Type: integer **Default:** PSIF_OEI

FZC_B_FILE

Type: integer **Default:** PSIF_OEI

FZC_FILE

Type: integer **Default:** PSIF_OEI

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double **Default:** 1e-14

IVO

Do ?

Type: boolean **Default:** false

J_FILE

Type: integer **Default:** 91

KEEP_J

Do keep half-transformed integrals?

Type: boolean **Default:** false

KEEP_PRESORT

Do ?

Type: boolean **Default:** false

LAGRAN_DOUBLE

Do ?

Type: boolean **Default:** false

LAGRAN_HALVE

Do ?

Type: boolean **Default:** false

LAG_IN_FILE

Type: integer **Default:** PSIF_MO_LAG

MAX_BUCKETS

Type: integer **Default:** 499

MODE

Possible Values: TO_MO, TO_AO

Type: string **Default:** TO_MO

MOORDER

Type: array **Default:** No Default

MP2R12A

Possible Values: MP2R12AERI, MP2R12AR12, MP2R12AR12T1

Type: string **Default:** MP2R12AERI

M_FILE

Type: integer **Default:** 0

OEI_A_FILE

Type: integer **Default:** PSIF_OEI

OEI_B_FILE

Type: integer **Default:** PSIF_OEI

OEI_FILE

Type: integer **Default:** PSIF_OEI

OPDM_IN_FILE

Type: integer **Default:** PSIF_MO_OPDM

OPDM_OUT_FILE

Type: integer **Default:** PSIF_AO_OPDM

PITZER

Do ?

Type: boolean **Default:** false

PRESORT_FILE

Type: integer **Default:** PSIF_SO_PRESORT

PRINT_LVL

Type: integer **Default:** 1

PRINT_MOS

Do ?

Type: boolean **Default:** false

PRINT_OE_INTEGRALS

Do ?

Type: boolean **Default:** false

PRINT_REORDER

Do ?

Type: boolean **Default:** false

PRINT_SORTED_OE_INTS

Do ?

Type: boolean **Default:** false

PRINT_SORTED_TE_INTS

Do ?

Type: boolean **Default:** false

PRINT_TE_INTEGRALS

Do ?

Type: boolean **Default:** false

PSIMRCC

Do ?

Type: boolean **Default:** false

QRHF

Do ?

Type: boolean **Default:** false

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

REORDER

Do ?

Type: boolean **Default:** false

RESTRICTED_DOCC

An array giving the number of restricted doubly-occupied orbitals per irrep (not excited in CI wavefunctions, but orbitals can be optimized in MCSCF)

Type: array **Default:** No Default

RESTRICTED_UOCC

An array giving the number of restricted unoccupied orbitals per irrep (not occupied in CI wavefunctions, but orbitals can be optimized in MCSCF)

Type: array **Default:** No Default

SORTED_TEI_FILE

Type: integer **Default:** PSIF_MO_TEI

SO_S_FILE

Type: integer **Default:** PSIF_OEI

SO_TEI_FILE

Type: integer **Default:** PSIF_SO_TEI

SO_T_FILE

Type: integer **Default:** PSIF_OEI

SO_V_FILE

Type: integer **Default:** PSIF_OEI

TPDM_ADD_REF

Do ?

Type: boolean **Default:** false

TPDM_FILE

Type: integer **Default:** PSIF_MO_TPDM

B.33 TRANSQT2

Performs transformations of integrals into the molecular orbital (MO) basis. This module is currently used by the (non-density fitted) MP2 and coupled cluster codes, but it is being phased out.

AO_BASIS

The algorithm to use for the $\langle VV||VV \rangle$ terms
Possible Values: NONE, DISK, DIRECT
Type: string **Default:** NONE

CACHELEVEL

Type: integer **Default:** 2

DELETE_TEI

Boolean to delete the SO-basis two-electron integral file after the transformation
Type: boolean **Default:** true

INTS_TOLERANCE

Minimum absolute value below which integrals are neglected. See the note at the beginning of Section B.

Type: double **Default:** 1e-14

PRINT_TEI

Do ?

Type: boolean **Default:** false

REFERENCE

Reference wavefunction type

Type: string **Default:** RHF

SEMICANONICAL

Convert ROHF MOs to semicanonical MOs

Type: boolean **Default:** true

C Expert Keywords Recognized by Each Module, for Advanced Users

C.1 GLOBALS

DEBUG

The amount of information to print to the output file

Type: integer **Default:** 0

DERTYPE

Derivative level

Possible Values: NONE, FIRST, SECOND, RESPONSE

Type: string **Default:** NONE

MAT_NUM_COLUMN_PRINT

Number of columns to print in calls to Matrix::print_mat

Type: integer **Default:** 5

WFN

Wavefunction type

Type: string **Default:** SCF

C.2 CCDENSITY

AEL

Do compute the approximate excitation level? See Stanton and Bartlett, JCP, 98, 1993, 7034.

Type: boolean **Default:** false

WFN

Wavefunction type

Type: string **Default:** SCF

XI_CONNECT

Do require $\bar{H}$ and R to be connected?

Type: boolean **Default:** false

C.3 CCENERGY

AO_BASIS

The algorithm to use for the $\langle VV||VV \rangle$ terms If AO_BASIS=NONE, the MO-basis integrals will be used; if AO_BASIS=DISK, the AO-basis integrals, stored on disk, will be used; if AO_BASIS=DIRECT, the AO-basis integrals will be computed on the fly as necessary. NB: The AO_BASIS=DIRECT option is not fully implemented and should only be used by experts. Default is NONE. Note: The developers recommend use of this keyword only as a last resort because it significantly slows the calculation. The current algorithms for handling the MO-basis four-virtual-index integrals have been significantly improved and are preferable to the AO-based approach.

Possible Values: NONE, DISK, DIRECT

Type: string **Default:** NONE

FORCE_RESTART

Do restart the coupled-cluster iterations even if MO phases are screwed up?

Type: boolean **Default:** false

WFN

Wavefunction type
Possible Values: CCSD, CCSD_T, EOM_CCSD, LEOM_CCSD, BCCD, BCCD_T, CC2, CC3, EOM_CC2, EOM_CC3, CCSD_MVD
Type: string **Default:** NONE

C.4 CCEOM

EXCITATION_RANGE

The depth into the occupied and valence spaces from which one-electron excitations are seeded into the Davidson guess to the CIS (the default of 2 includes all single excitations between HOMO-1, HOMO, LUMO, and LUMO+1). This CIS is in turn the Davidson guess to the EOM-CC. Expand to capture more exotic excited states in the EOM-CC calculation
Type: integer **Default:** 2

WFN

Wavefunction type
Possible Values: EOM_CCSD, EOM_CC2, EOM_CC3
Type: string **Default:** EOM_CCSD

C.5 CCHBAR

WFN

Wavefunction type
Type: string **Default:** SCF

C.6 CCLAMBDA

JOBTYPE

Type of job being performed
Type: string **Default:** No Default

WFN

Wavefunction type
Type: string **Default:** SCF

C.7 CCRESPONSE

WFN

Wavefunction type

Type: string **Default:** SCF

C.8 CCSORT

WFN

Wavefunction type

Type: string **Default:** No Default

C.9 CCTRIPLES

WFN

Wavefunction type

Type: string **Default:** SCF

C.10 CIS

WFN

Wavefunction type
Possible Values: CCSD, CCSD_T, EOM_CCSD, CIS
Type: string **Default:** CIS

C.11 CLAG

WFN

Wavefunction type

Type: string **Default:** NONE

C.12 DETCI

BENDAZZOLI

Do use some routines based on the papers of Bendazzoli et al. to calculate sigma? Seems to be slower and not worthwhile; may disappear eventually. Works only for full CI and I don't remember if I could see how their clever scheme might be extended to RAS in general.
Type: boolean **Default:** false

CC_FIX_EXTERNAL

Do fix amplitudes involving RAS I or RAS IV? Useful in mixed MP2-CC methods.

Type: boolean **Default:** false

CC_FIX_EXTERNAL_MIN

Number of external indices before amplitude gets fixed by CC_FIX_EXTERNAL. Experimental.

Type: integer **Default:** 1

CC_MACRO

CC_MACRO = [[ex_lvl, max_holes_I, max_parts_IV, max_I+IV], [ex_lvl, max_holes_I, max_parts_IV, max_I+IV], ...] Optional additional restrictions on allowed exictations in coupled-cluster computations, based on macroconfiguration selection. For each sub-array, [ex_lvl, max_holes_I, max_parts_IV, max_I+IV], eliminate cluster amplitudes in which: [the excitation level (holes in I + II) is equal to ex_lvl] AND [there are more than max_holes_I holes in RAS I, there are more than max_parts_IV particles in RAS IV, OR there are more than max_I+IV quasiparticles in RAS I + RAS IV].

Type: array **Default:** No Default

CC_MIXED

Do ignore block if num holes in RAS I and II is $>$ cc_ex_lvl and if any indices correspond to RAS I or IV (i.e., include only all-active higher excitations)?

Type: boolean **Default:** true

CC_UPDATE_EPS

Do update T amplitudes with orbital eigenvalues? (Usually would do this). Not doing this is experimental.

Type: boolean **Default:** true

CC_VARIATIONAL

Do use variational energy expression in CC computation? Experimental.

Type: boolean **Default:** false

EX_ALLOW

An array of length EX_LEVEL specifying whether each excitation type (S,D,T, etc.) is allowed (1 is allowed, 0 is disallowed). Used to specify non-standard CI spaces such as CIST.

Type: array **Default:** No Default

FCI_STRINGS

Do store strings specifically for FCI? (Defaults to TRUE for FCI.)

Type: boolean **Default:** false

FILTER_GUESS

Do invoke the FILTER_GUESS options that are used to filter out some trial vectors which may not have the appropriate phase convention between two determinants? This is useful to remove, e.g., delta states when a sigma state is desired. The user inputs two determinants (by giving the absolute alpha string number and beta string number for each), and also the desired phase between these two determinants for guesses which are to be kept. FILTER_GUESS = TRUE turns on the filtering routine. Requires additional keywords FILTER_GUESS_DET1, FILTER_GUESS_DET2, and FILTER_GUESS_SIGN.

Type: boolean **Default:** false

FILTER_GUESS_DET1

Array specifying the absolute alpha string number and beta string number for the first determinant in the filter procedure. (See FILTER_GUESS).

Type: array **Default:** No Default

FILTER_GUESS_DET2

Array specifying the absolute alpha string number and beta string number for the second determinant in the filter procedure. (See FILTER_GUESS).

Type: array **Default:** No Default

FILTER_GUESS_SIGN

The required phase (1 or -1) between the two determinants specified by FILTER_GUESS_DET1 and FILTER_GUESS_DET2

Type: integer **Default:** 1

FILTER_ZERO_DET

If present, the code will try to filter out a particular determinant by setting its CI coefficient to zero. FILTER_ZERO_DET = (alphastr betastr) specifies the absolute alpha and beta string numbers of the target determinant. This could be useful for trying to exclude states that have a nonzero CI coefficient for the given determinant. However, this option was experimental and may not be effective.

Type: array **Default:** No Default

FOLLOW_VECTOR

In following a particular root (see ROOT keyword), sometimes the root number changes. To follow a root of a particular character, one can specify a list of determinants and their coefficients, and the code will follow the root with the closest overlap. The user specifies arrays containing the absolute alpha string indices (A_i below), absolute beta indices (B_i below), and CI coefficients (C_i below) to form the desired vector. FOLLOW_VECTOR_ALPHAS specifies the alpha string indices. The format is FOLLOW_VECTOR = [[[A_1, B_1], C_1], [[A_2, B_2], C_2], ...].

Type: array **Default:** No Default

GUESS_VECTOR

Guess vector type. Accepted values are UNIT for a unit vector guess (NUM_ROOTS and NUM_INIT_VECS must both be 1); H0_BLOCK to use eigenvectors from the H0 BLOCK submatrix (default); DFILE to use NUM_ROOTS previously converged vectors in the D file; IMPORT to import a guess previously exported from a CI computation (possibly using a different CI space)

Possible Values: UNIT, H0_BLOCK, DFILE, IMPORT

Type: string **Default:** H0_BLOCK

H0_BLOCKSIZE

This parameter specifies the size of the H0 block of the Hamiltonian which is solved exactly. The n determinants with the lowest SCF energy are selected, and a submatrix of the Hamiltonian is formed using these determinants. This submatrix is used to accelerate convergence of the CI iterations in the BOLSEN and MITRUSHENKOV iteration schemes, and also to find a good starting guess for the SEM method if GUESS_VECTOR = H0_BLOCK. Defaults to 400. Note that the program may change the given size for Ms=0 cases (Ms0 = TRUE) if it determines that the H0 block includes only one member of a pair of determinants related by time reversal symmetry. For very small block sizes, this could conceivably eliminate the entire H0 block; the program should print warnings if this occurs.

Type: integer **Default:** 400

H0_BLOCK_COUPLING

Do use coupling block in preconditioner?

Type: boolean **Default:** false

H0_BLOCK_COUPLING_SIZE

Parameters which specifies the size of the coupling block within the generalized davidson preconditioner.

Type: integer **Default:** 0

H0_GUESS_SIZE

size of H0 block for initial guess

Type: integer **Default:** 400

HD_AVG

How to average H diag energies over spin coupling sets. HD_EXACT uses the exact diagonal energies which results in expansion vectors which break spin symmetry. HD_KAVE averages the diagonal energies over a spin-coupling set yielding spin pure expansion vectors. ORB_ENER employs the sum of orbital energy approximation giving spin pure expansion vectors but usually doubles the number of Davidson iterations. EVANGELISTI uses the sums and differences of orbital energies with the SCF reference energy to produce spin pure expansion vectors. LEININGER approximation which subtracts the one-electron contribution from the orbital energies, multiplies by 0.5, and adds the one-electron contribution back in, producing spin pure expansion vectors and developed by Matt Leininger and works as well as EVANGELISTI.

Type: string **Default:** EVANGELISTI

HD_OTF

Do compute the diagonal elements of the Hamiltonian matrix on-the-fly? Otherwise, a diagonal element vector is written to a separate file on disk.

Type: boolean **Default:** true

MIXED

Do allow 'mixed' RAS II/RAS III excitations into the CI space? If FALSE, then if there are any electrons in RAS III, then the number of holes in RAS I cannot exceed the given excitation level EX_LEVEL.

Type: boolean **Default:** true

MIXED4

Do allow 'mixed' excitations involving RAS IV into the CI space. Useful to specify a split-virtual CISD[TQ] computation. If FALSE, then if there are any electrons in RAS IV, then the number of holes in RAS I cannot exceed the given excitation level EX_LEVEL.

Type: boolean **Default:** true

MPN_ORDER_SAVE

If 0, save the MPn energy; if 1, save the MP(2n-1) energy (if available from MPN_WIGNER=true); if 2, save the MP(2n-2) energy (if available from MPN_WIGNER=true).

Type: integer **Default:** 0

MPN_SCHMIDT

Do employ an orthonormal vector space rather than storing the kth order wavefunction?

Type: boolean **Default:** false

MPN_WIGNER

Do use Wigner formulas in the Empn series?

Type: boolean **Default:** true

NO_DFILE

Do use the last vector space in the BVEC file to write scratch DVEC rather than using a separate DVEC file? (Only possible if NUM_ROOTS = 1.)

Type: boolean **Default:** false

NUM_INIT_VECS

The number of initial vectors to use in the CI iterative procedure. Defaults to the number of roots.

Type: integer **Default:** 0

OPDM_KE

Do compute the kinetic energy contribution from the correlated part of the one-particle density matrix?

Type: boolean **Default:** false

PERTURB_MAGNITUDE

z in $H = H_0 + zH_1$

Type: double **Default:** 1.0

R4S

Do restrict strings with $e-$ in RAS IV? Useful to reduce the number of strings required if MIXED4=true, as in a split-virutal CISD[TQ] computation. If more than one electron is in RAS IV, then the holes in RAS I cannot exceed the number of particles in RAS III + RAS IV (i.e., EX_LEVEL), or else the string is discarded.

Type: boolean **Default:** false

RAS1

An array giving the number of orbitals per irrep for RAS1

Type: array **Default:** No Default

RAS2

An array giving the number of orbitals per irrep for RAS2

Type: array **Default:** No Default

RAS3

An array giving the number of orbitals per irrep for RAS3

Type: array **Default:** No Default

RAS4

An array giving the number of orbitals per irrep for RAS4

Type: array **Default:** No Default

REFERENCE_SYM

Irrep for CI vectors; -1 = find automatically. This option allows the user to look for CI vectors of a different irrep than the reference. This probably only makes sense for Full CI, and it would probably not work with unit vector guesses. Numbering starts from zero for the totally-symmetric irrep.

Type: integer **Default:** -1

REPL_OTF

Do string replacements on the fly in DETCI? Can save a gigantic amount of memory (especially for truncated CI's) but is somewhat flaky and hasn't been tested for a while. It may work only works for certain classes of RAS calculations. The current code is very slow with this option turned on.

Type: boolean **Default:** false

SF_RESTRICT

Do eliminate determinants not valid for spin-complete spin-flip CI's? [see J. S. Sears et al, J. Chem. Phys. 118, 9084-9094 (2003)]

Type: boolean **Default:** false

SIGMA_OVERLAP

Do print the sigma overlap matrix? Not generally useful.

Type: boolean **Default:** false

WFN

Wavefunction type

Possible Values: DETCI, CI, ZAPTN, DETCAS, CASSCF, RASSCF

Type: string **Default:** DETCI

C.13 DFMP2

DF_INTS_IO

IO caching for CP corrections, etc
Possible Values: NONE, SAVE, LOAD
Type: string **Default:** NONE

MADMP2_SLEEP

A helpful option, used only in debugging the MADNESS version
Type: integer **Default:** 0

C.14 LMP2

WFN

Wavefunction type
Type: string **Default:** LMP2

C.15 MCSCF

ROTATE_MO_ANGLE

For orbital rotations after convergence, the angle (in degrees) by which to rotate.
Type: integer **Default:** 0

ROTATE_MO_IRREP

For orbital rotations after convergence, irrep (1-based, Cotton order) of the orbitals to rotate.
Type: integer **Default:** 1

ROTATE_MO_P

For orbital rotations after convergence, number of the first orbital (1-based) to rotate.
Type: integer **Default:** 1

ROTATE_MO_Q

For orbital rotations after convergence, number of the second orbital (1-based) to rotate.
Type: integer **Default:** 2

C.16 MP2

JOBTYPE

Type of job being performed

Type: string **Default:** SP

WFN

Wavefunction type

Possible Values: MP2

Type: string **Default:** MP2

C.17 MRCC

MRCC_METHOD

If more than one root is requested and calc=1, LR-CC (EOM-CC) calculation is performed automatically for the excited states. This overrides all automatic determination of method and will only work with `energy()`. This becomes CC/CI (option #5) in fort.56

Value	Method	Description
1	CC	
2	CC(n-1)[n]	
3	CC(n-1)(n)	(CC(n-1)[n] energy is also calculated)
4	CC(n-1)(n)_L	(CC(n-1)[n] and CC(n-1)(n) energies are also calculated)
5	CC(n)-1a	
6	CC(n)-1b	
7	CCn	
8	CC(n)-3	

Type: integer **Default:** 1

MRCC_OMP_NUM_THREADS

Sets the OMP_NUM_THREADS environment variable before calling MRCC. If the environment variable OMP_NUM_THREADS is set prior to calling PSI4 then that value is used. When set, this option overrides everything. Be aware the `-n` command-line option described in section 3.9 does not affect MRCC.

Type: integer **Default:** 1

MRCC_RESTART

The program restarts from the previously calculated parameters if it is 1. In case it is 2, the program executes automatically the lower-level calculations of the same type consecutively (e.g., CCSD, CCSDT, and CCSDTQ if CCSDTQ is requested) and restarts each calculation from the previous one (rest=2 is available only for energy calculations). Currently, only a value of 0 and 2 are supported. This becomes `rest` (option #4) in fort.56.

Type: integer **Default:** 0

C.18 PLUGIN-CCSD-SERIAL

C.19 PSIMRCC

PERTURB_CBS

Do compute the perturbative corrections for basis set incompleteness?
Type: boolean **Default:** false

PERTURB_CBS_COUPLING

Do include the terms that couple different reference determinants in perturbative CBS correction computations?
Type: boolean **Default:** true

TIKHONOW_TRIPLES

Do use Tikhonow regularization in (T) computations?
Type: boolean **Default:** false

USE_SPIN_SYMMETRY

Whether to use spin symmetry to map equivalent configurations onto each other, for efficiency
Type: boolean **Default:** true

C.20 SAPT

DO_THIRD_ORDER

Do compute third-order corrections?
Type: boolean **Default:** false

C.21 SCF

C.21.1 DFSCF Algorithm

DF_FITTING_CONDITION

Fitting Condition
Type: integer **Default:** 1

DF_INTS_IO

IO caching for CP corrections, etc
Possible Values: NONE, SAVE, LOAD
Type: string **Default:** NONE

C.21.2 Environmental Effects

DISTRIBUTED_MATRIX

The dimension sizes of the distributed matrix
Type: array **Default:** No Default

PARALLEL

Do run in parallel?
Type: boolean **Default:** false

TILE_SZ

The tile size for the distributed matrices
Type: integer **Default:** 512

C.21.3 General Wavefunction Info

WFN

Wavefunction type
Possible Values: SCF
Type: string **Default:** SCF

C.21.4 Misc.

SAPT

Are going to do SAPT? If so, what part?
Possible Values: FALSE, 2-DIMER, 2-MONOMER_A, 2-MONOMER_B, 3-TRIMER, 3-DIMER_AB, 3-DIMER_BC, 3-DIMER_AC, 3-MONOMER_A, 3-MONOMER_B, 3-MONOMER_C
Type: string **Default:** FALSE

C.21.5 SAD Guess Algorithm

SAD_CHOL_TOLERANCE

SAD Guess Cholesky Cutoff (for eliminating redundancies). See the note at the beginning of Section B.
Type: double **Default:** 1e-7

SAD_D_CONVERGENCE

Convergence criterion for SCF density in SAD Guess. See the note at the beginning of Section B.
Type: double **Default:** 1e-5

SAD_E_CONVERGENCE

Convergence criterion for SCF energy in SAD Guess. See the note at the beginning of Section B.

Type: double **Default:** 1e-5

SAD_F_MIX_START

SAD Guess F-mix Iteration Start

Type: integer **Default:** 50

SAD_MAXITER

Maximum number of SAD guess iterations

Type: integer **Default:** 50

SAD_PRINT

The amount of SAD information to print to the output

Type: integer **Default:** 0

C.22 TRANSQT

RAS1

An array giving the number of orbitals per irrep for RAS1

Type: array **Default:** No Default

RAS2

An array giving the number of orbitals per irrep for RAS2

Type: array **Default:** No Default

RAS3

An array giving the number of orbitals per irrep for RAS3

Type: array **Default:** No Default

RAS4

An array giving the number of orbitals per irrep for RAS4

Type: array **Default:** No Default

WFN

Wavefunction type

Type: string **Default:** CCSD

C.23 TRANSQT2

WFN

Wavefunction type

Type: string **Default:** No Default

D `PSI` Variables Set by Each Module

This listing provides details of the `PSI` variables that can be set by each module. Italicized portions of the variable name vary by root number, calculation order, etc. Note that neither properties variables nor variables set in the psi driver are yet incorporated into this documentation.

D.1 GLOBALS

D.2 ADC

CURRENT ENRGY

D.3 CCDENSITY

D.4 CCENERGY

BRUECKNER CONVERGED
CC CORRELATION ENERGY
CC TOTAL ENERGY
CC2 CORRELATION ENERGY
CC2 TOTAL ENERGY
CC3 CORRELATION ENERGY
CC3 TOTAL ENERGY
CCSD CORRELATION ENERGY
CCSD OPPOSITE-SPIN CORRELATION ENERGY
CCSD SAME-SPIN CORRELATION ENERGY
CCSD TOTAL ENERGY
CURRENT CORRELATION ENERGY
CURRENT ENERGY
LCC2 (+LMP2) TOTAL ENERGY
LCCSD (+LMP2) TOTAL ENERGY
MP2 CORRELATION ENERGY
MP2 OPPOSITE-SPIN CORRELATION ENERGY

MP2 SAME-SPIN CORRELATION ENERGY
MP2 TOTAL ENERGY
SCF TOTAL ENERGY
SCF TOTAL ENERGY (CHKPT)
SCS-CCSD CORRELATION ENERGY
SCS-CCSD OPPOSITE-SPIN CORRELATION ENERGY
SCS-CCSD SAME-SPIN CORRELATION ENERGY
SCS-CCSD TOTAL ENERGY
SCS-MP2 CORRELATION ENERGY
SCS-MP2 OPPOSITE-SPIN CORRELATION ENERGY
SCS-MP2 SAME-SPIN CORRELATION ENERGY
SCS-MP2 TOTAL ENERGY
SCSN-MP2 CORRELATION ENERGY
SCSN-MP2 OPPOSITE-SPIN CORRELATION ENERGY
SCSN-MP2 SAME-SPIN CORRELATION ENERGY
SCSN-MP2 TOTAL ENERGY

D.5 CCEOM

CC ROOT N TOTAL ENERGY
CURRENT CORRELATION ENERGY
CURRENT ENERGY

D.6 CCHBAR

D.7 CCLAMBDA

D.8 CCRESPONSE

D.9 CCSORT

D.10 CCTRIPLES

(T) CORRECTION ENERGY
AAA (T) CORRECTION ENERGY
AAB (T) CORRECTION ENERGY
ABB (T) CORRECTION ENERGY
BBB (T) CORRECTION ENERGY
CCSD(T) CORRELATION ENERGY
CCSD(T) TOTAL ENERGY
CURRENT CORRELATION ENERGY
CURRENT ENERGY

D.11 CIS

D.12 CLAG

D.13 CPHF

D.14 DCFT

CURRENT ENERGY
DCFT ENERGY
DCFT LAMBDA ENERGY
DCFT SCF ENERGY
DCFT TAU SQUARED CORRECTION
MP2 ENERGY

D.15 DETCI

CI CORRELATION ENERGY
CI DIPOLE X
CI DIPOLE Y
CI DIPOLE Z
CI QUADRUPOLE XX
CI QUADRUPOLE XY
CI QUADRUPOLE XZ
CI QUADRUPOLE YY
CI QUADRUPOLE YZ
CI QUADRUPOLE ZZ
CI ROOT N $\rightarrow$ ROOT M DIPOLE X, for X = X, Y, Z
CI ROOT N $\rightarrow$ ROOT M QUADRUPOLE X, for X = XX, XY, XZ, YY, YZ, ZZ
CI ROOT N CORRELATION ENERGY
CI ROOT N DIPOLE X, for X = X, Y, Z
CI ROOT N QUADRUPOLE X, for X = XX, XY, XZ, YY, YZ, ZZ
CI ROOT N TOTAL ENERGY
CI STATE-AVERAGED CORRELATION ENERGY
CI STATE-AVERAGED TOTAL ENERGY
CI TOTAL ENERGY
CIN CORRELATION ENERGY
CIN TOTAL ENERGY
CISD CORRELATION ENERGY
CISD TOTAL ENERGY
CISDT CORRELATION ENERGY
CISDT TOTAL ENERGY
CISDTQ CORRELATION ENERGY
CISDTQ TOTAL ENERGY
CURRENT CORRELATION ENERGY
CURRENT ENERGY
CURRENT REFERENCE ENERGY
FCI CORRELATION ENERGY
FCI TOTAL ENERGY
MPN CORRELATION ENERGY
MPN TOTAL ENERGY

ZAPTN CORRELATION ENERGY
ZAPTN TOTAL ENERGY

D.16 DFCC

CURRENT ENERGY
RPA DELTA ENERGY
RPA DRPA ENERGY
RPA MP2J ENERGY
RPA SCALED DELTA ENERGY
RPA TOTAL ENERGY

D.17 DFMP2

CURRENT CORRELATION ENERGY
CURRENT ENERGY
DF-MP2 CORRELATION ENERGY
DF-MP2 OPPOSITE-SPIN ENERGY
DF-MP2 SAME-SPIN ENERGY
DF-MP2 SINGLES ENERGY
DF-MP2 TOTAL ENERGY
SCS-DF-MP2 CORRELATION ENERGY
SCS-DF-MP2 TOTAL ENERGY

D.18 FINDIF

D.19 LMP2

CURRENT ENERGY
LMP2 ENERGY

D.20 MCSCF

CURRENT ENERGY
CURRENT REFERENCE ENERGY
MCSCF TOTAL ENERGY
SCF TOTAL ENERGY

D.21 MINTS

D.22 MP2

CURRENT CORRELATION ENERGY
CURRENT ENERGY
MP2 CORRELATION ENERGY
MP2 OPPOSITE-SPIN CORRELATION ENERGY
MP2 SAME-SPIN CORRELATION ENERGY
MP2 TOTAL ENERGY

SCS-MP2 CORRELATION ENERGY
SCS-MP2 OPPOSITE-SPIN CORRELATION ENERGY
SCS-MP2 SAME-SPIN CORRELATION ENERGY
SCS-MP2 TOTAL ENERGY

D.23 MRCC

D.24 OMP2

D.25 OPTKING

CURRENT ENERGY
OPTIMIZATION ITERATIONS

D.26 PLUGIN-CCSD-SERIAL

D.27 PSIMRCC

CURRENT ENERGY
MRCC TOTAL ENERGY

D.28 RESPONSE

D.29 SAPT

CURRENT ENERGY
MP2C DELTA MP2C ENERGY
MP2C DIMER MP2 ENERGY
MP2C MONOMER A MP2 ENERGY
MP2C MONOMER B MP2 ENERGY
MP2C MP2 ENERGY
MP2C MP2C ENERGY
MP2C TDDFT ENERGY
MP2C UCHF ENERGY
SAPT CT ENERGY
SAPT DISP ENERGY
SAPT ELST ENERGY
SAPT ENERGY
SAPT EXCH ENERGY
SAPT IND ENERGY
SAPT SAPT0 ENERGY
SAPT SAPT2 ENERGY
SAPT SAPT2+ ENERGY
SAPT SAPT2+(3) ENERGY
SAPT SAPT2+3 ENERGY
SAPT SCS-DISP ENERGY
SAPT SCS-SAPT0 ENERGY

D.30 SCF

CURRENT ENERGY
CURRENT REFERENCE ENERGY
SCF ITERATION ENERGY
SCF TOTAL ENERGY

D.31 STABLE

D.32 TRANSQT

D.33 TRANSQT2

E The Test Suite and Sample Inputs

PSI4 is distributed with an extensive test suite, which can be found in $PSI4/tests. After building the source code, these can automatically be run by running `make tests` in the complilation directory. Sample input files can be found in the the samples subdirectory of the top-level Psi directory. The samples provided, and corresponding location in the samples folder, are:-

adc1	ADC/6-31G** on H_2O
adc2	ADC/aug-cc-pVDZ on two water molecules that are distant from 1000 angstroms from each other
castup1	Test of SAD/Cast-up (mainly not dying due to file weirdness)
cc1	RHF-CCSD 6-31G** all-electron optimization of the H2O molecule
cc10	ROHF-CCSD cc-pVDZ energy for the $^2\Sigma^+$ state of the CN radical
cc11	Frozen-core CCSD(ROHF)/cc-pVDZ on CN radical with disk-based AO algorithm
cc12	Single point energies of multiple excited states with EOM-CCSD
cc13	UHF-CCSD/cc-pVDZ 3B1 CH2 geometry optimization via analytic gradients
cc13a	UHF-CCSD(T)/cc-pVDZ 3B1 CH2 geometry optimization via analytic gradients

cc14	ROHF-CCSD/cc-pVDZ 3B1 CH2 geometry optimization via analytic gradients
cc15	RHF-B-CCD(T)/6-31G** H2O single-point energy (fzc, MO-basis $\langle ab\|cd\rangle$)
cc16	UHF-B-CCD(T)/cc-pVDZ 3B1 CH2 single-point energy (fzc, MO-basis $\langle ab\|cd\rangle$)
cc17	Single point energies of multiple excited states with EOM-CCSD
cc18	RHF-CCSD-LR/cc-pVDZ static polarizability of HOF
cc19	CCSD/cc-pVDZ dipole polarizability at two frequencies
cc2	6-31G** H2O CCSD optimization by energies, with Z-Matrix input
cc22	ROHF-EOM-CCSD/DZ on the lowest two states of each irrep in $^3B_1$ CH_2.
cc23	ROHF-EOM-CCSD/DZ analytic gradient lowest 2B1 state of H2O+ (A1 excitation)
cc24	Single point gradient of 1-2B1 state of H2O+ with EOM-CCSD
cc25	Single point gradient of 1-2B2 state of H2O+ with EOM-CCSD
cc26	Single-point gradient, analytic and via finite-differences of 2-1A1 state of H2O with EOM-CCSD
cc27	Single point gradient of 1-1B2 state of H2O with EOM-CCSD
cc28	CCSD/cc-pVDZ optical rotation calculation (length gauge only) on Z-mat H_2O_2
cc29	CCSD/cc-pVDZ optical rotation calculation (both gauges) on Cartesian H_2O_2
cc3	cc3: RHF-CCSD/6-31G** H2O geometry optimization and vibrational frequency analysis by finite-differences of gradients
cc30	CCSD/sto-3g optical rotation calculation (length gauge only) at two frequencies on methyloxirane

cc31	CCSD/sto-3g optical rotation calculation (both gauges) at two frequencies on methyloxirane
cc32	CC3/cc-pVDZ H_2O R_e geom from Olsen et al., JCP 104, 8007 (1996)
cc33	CC3(UHF)/cc-pVDZ H_2O R_e geom from Olsen et al., JCP 104, 8007 (1996)
cc34	RHF-CCSD/cc-pVDZ energy of H2O partitioned into pair energy contributions.
cc35	CC3(ROHF)/cc-pVDZ H_2O R_e geom from Olsen et al., JCP 104, 8007 (1996)
cc36	CC2(RHF)/cc-pVDZ energy of H2O.
cc37	CC2(UHF)/cc-pVDZ energy of H2O+.
cc38	RHF-CC2-LR/cc-pVDZ static polarizabilities of HOF molecule.
cc39	RHF-CC2-LR/cc-pVDZ dynamic polarizabilities of HOF molecule.
cc4	RHF-CCSD(T) cc-pVQZ frozen-core energy of the BH molecule, with Cartesian input. After the computation, the checkpoint file is renamed, using the PSIO handler.
cc40	RHF-CC2-LR/cc-pVDZ optical rotation of H2O2. gauge = length, omega = (589 355 nm)
cc41	RHF-CC2-LR/cc-pVDZ optical rotation of H2O2. gauge = both, omega = (589 355 nm)
cc42	RHF-CC2-LR/STO-3G optical rotation of (S)-methyloxirane. gauge = length, omega = (589 355 nm)
cc43	RHF-CC2-LR/STO-3G optical rotation of (S)-methyloxirane. gauge = both, omega = (589 355 nm)
cc44	Test case for some of the PSI4 out-of-core codes. The code is given only 2.0 MB of memory, which is insufficient to hold either the A1 or B2 blocks of an ovvv quantity in-core, but is sufficient to hold at least two copies of an oovv quantity in-core.

cc45	RHF-EOM-CC2/cc-pVDZ lowest two states of each symmetry of H2O.
cc46	EOM-CC2/cc-pVDZ on H_2O_2 with two excited states in each irrep
cc49	EOM-CC3(UHF) on CH radical with user-specified basis and properties for particular root
cc4a	RHF-CCSD(T) cc-pVQZ frozen-core energy of the BH molecule, with Cartesian input. This version tests the FROZEN_DOCC option explicitly
cc5	RHF CCSD(T) aug-cc-pvtz frozen-core energy of C4NH4 Anion
cc50	EOM-CC3(ROHF) on CH radical with user-specified basis and properties for particular root
cc51	EOM-CC3/cc-pVTZ on H_2O
cc52	CCSD Response for H2O2
cc5a	RHF CCSD(T) STO-3G frozen-core energy of C4NH4 Anion
cc6	Frozen-core CCSD(T)/cc-pVDZ on C_4H_4N anion with disk ao algorithm
cc8	UHF-CCSD(T) cc-pVDZ frozen-core energy for the $^2\Sigma^+$ state of the CN radical, with Z-matrix input.
cc8a	ROHF-CCSD(T) cc-pVDZ frozen-core energy for the $^2\Sigma^+$ state of the CN radical, with Cartesian input.
cc8b	ROHF-CCSD cc-pVDZ frozen-core energy for the $^2\Sigma^+$ state of the CN radical, with Cartesian input.
cc8c	ROHF-CCSD cc-pVDZ frozen-core energy for the $^2\Sigma^+$ state of the CN radical, with Cartesian input.
cc9	UHF-CCSD(T) cc-pVDZ frozen-core energy for the $^2\Sigma^+$ state of the CN radical, with Z-matrix input.
cc9a	ROHF-CCSD(T) cc-pVDZ energy for the $^2\Sigma^+$ state of the CN radical, with Z-matrix input.
cisd-h2o+-0	6-31G** H2O+ Test CISD Energy Point

cisd-h2o+-1	6-31G** H2O+ Test CISD Energy Point
cisd-h2o+-2	6-31G** H2O+ Test CISD Energy Point
cisd-h2o-clpse	6-31G** H2O Test CISD Energy Point with subspace collapse
cisd-opt-fd	H2O CISD/6-31G** Optimize Geometry by Energies
cisd-sp	6-31G** H2O Test CISD Energy Point
cisd-sp-2	6-31G** H2O Test CISD Energy Point
dcft1	DCFT calculation for the He dimer, with the K06 functional. This performs a simultaneous update of the orbitals and cumulant, using DIIS extrapolation. Four-virtual integrals are handled in the MO Basis.
dcft2	DCFT calculation for the He dimer, with the K06 functional. This performs a two-step update of the orbitals and cumulant, using DIIS extrapolation. Four-virtual integrals are handled in the MO Basis.
dcft3	DCFT calculation for the He dimer, with the K06 functional. This performs a simultaneous update of the orbitals and cumulant, using DIIS extrapolation. Four-virtual integrals are handled in the AO Basis, using integrals stored on disk.
dcft4	DCFT calculation for the Ne atom, with the K06 functional. This performs both two-step and simultaneous update of the orbitals and cumulant using DIIS extrapolation. Four-virtual integrals are handled in the MO Basis. The reference DCFT energy is taken from the JCP 133 174122 (2010) paper
dcft5	DCFT calculation for the Ne+ ion (doublet ground state). This performs both two-step and simultaneous update of the orbitals and cumulant using DIIS extrapolation. Four-virtual integrals are handled in the MO Basis.
dfmp2_1	Density fitted MP2 cc-PVDZ/cc-pVDZ-RI computation of formic acid dimer binding energy using automatic counterpoise correction. Monomers are specified using Cartesian coordinates.
dfmp2_2	Density fitted MP2 energy of H2, using density fitted reference and automatic looping over cc-pVDZ and cc-pVTZ basis sets. Results are tabulated using the built in table functions by using the default options and by specifiying the format.

dfscf-bz2	Benzene Dimer DF-HF/cc-pVDZ
dft1	DFT Functional Test
dft2	DFT Functional Test
fci-dipole	6-31G H2O Test FCI Energy Point
fci-h2o	6-31G H2O Test FCI Energy Point
fci-h2o-2	6-31G H2O Test FCI Energy Point
fci-h2o-fzcv	6-31G H2O Test FCI Energy Point
fci-tdm	He2+ FCI/cc-pVDZ Transition Dipole Moment
fci-tdm-2	BH-H2+ FCI/cc-pVDZ Transition Dipole Moment
fd-freq-energy	SCF STO-3G finite-difference frequencies from energies
fd-freq-energy-large	SCF DZ finite difference frequencies by energies for C4NH4
fd-freq-gradient	STO-3G frequencies for H2O by finite-differences of gradients
fd-freq-gradient-large	SCF DZ finite difference frequencies by energies for C4NH4
fd-gradient	SCF STO-3G finite-difference tests
frac	Carbon/UHF Fractionally-Occupied SCF Test Case
matrix1	An example of using BLAS and LAPACK calls directly from the Psi input file, demonstrating matrix multiplication, eigendecomposition, Cholesky decomposition and LU decomposition. These operations are performed on vectors and matrices provided from the Psi library.
mcscf1	ROHF 6-31G** energy of the $^3B_1$ state of CH_2, with Z-matrix input. The occupations are specified explicitly.
mcscf2	TCSCF cc-pVDZ energy of asymmetrically displaced ozone, with Z-matrix input.

mcscf3	RHF 6-31G** energy of water, using the MCSCF module and Z-matrix input.
mints1	Symmetry tests for a range of molecules. This doesn't actually compute any energies, but serves as an example of the many ways to specify geometries in Psi4.
mints2	A test of the basis specification. A benzene atom is defined using a ZMatrix containing dummy atoms and various basis sets are assigned to different atoms. The symmetry of the molecule is automatically lowered to account for the different basis sets.
mints3	Test individual integral objects for correctness.
mints4	A demonstration of mixed Cartesian/ZMatrix geometry specification, using variables, for the benzene-hydronium complex. Atoms can be placed using ZMatrix coordinates, whether they belong to the same fragment or not. Note that the Cartesian specification must come before the ZMatrix entries because the former define absolute positions, while the latter are relative.
mom	Maximum Overlap Method (MOM) Test. MOM is designed to stabilize SCF convergence and to target excited Slater determinants directly.
mp2_1	All-electron MP2 6-31G** geometry optimization of water
mpn-bh	MP(n)/aug-cc-pVDZ BH Energy Point, with n=2-19. Compare against M. L. Leininger et al., J. Chem. Phys. 112, 9213 (2000)
mrcc1	CCSDT cc-pVDZ energy for the H2O molecule using MRCC
mrcc2	CCSDT(Q) cc-pVDZ energy for the H2O molecule using MRCC. This example builds up from CCSD. First CCSD, then CCSDT, finally CCSDT(Q).
mrcc3	CCSD(T) cc-pVDZ geometry optimization for the H2O molecule using MRCC.
mrcc4	CCSDT cc-pVDZ optimization and frequencies for the H2O molecule using MRCC
opt1	SCF STO-3G geometry optimzation, with Z-matrix input

opt1-fd	SCF STO-3G geometry optimzation, with Z-matrix input, by finite-differences
opt2	SCF DZ allene geometry optimzation, with Cartesian input
opt2-fd	SCF DZ allene geometry optimzation, with Cartesian input
opt3	SCF cc-pVDZ geometry optimzation, with Z-matrix input
opt4	SCF cc-pVTZ geometry optimzation, with Z-matrix input
opt5	6-31G** UHF CH2 3B1 optimization
plugin_df_ccsd	Test DF-CCSD(T) energy for H2O/aug-cc-pVDZ
plugin_gpu_ccsd	cc-pvdz H2O Test CCSD energies
plugin_libcim	sto-3g (H2O)8 Test Boys localization
props1	RHF STO-3G dipole moment computation, performed by applying a finite electric field and numerical differentiation.
props2	DF-SCF cc-pVDZ of benzene-hydronium ion, scanning the dissociation coordinate with Python's built-in loop mechanism. The geometry is specified by a Z-matrix with dummy atoms, fixed parameters, updated parameters, and separate charge/multiplicity specifiers for each monomer. One-electron properties computed for dimer and one monomer.
psimrcc-sp1	Mk-MRCCSD single point. ${}^3\Sigma^- O_2$ state described using the Ms = 0 component of the triplet. Uses ROHF triplet orbitals.
pubchem1	Benzene vertical singlet-triplet energy difference computation, using the PubChem database to obtain the initial geometry, at the UHF an ROHF levels of theory.
pywrap_alias	Test parsed and exotic calls to energy() like zapt4, mp2.5, and cisd are working
pywrap_all	Intercalls among python wrappers- database, cbs, optimize, energy, etc. Though each call below functions individually, running them all in sequence or mixing up the sequence is aspirational at present. Also aspirational is using the intended types of gradients.

pywrap_cbs1	Various basis set extrapolation tests
pywrap_db1	Database calculation, so no molecule section in input file. Portions of the full databases, restricted by subset keyword, are computed by sapt0 and dfmp2 methods.
pywrap_db2	Database calculation with psi4-generated input. Should not be used as a model input file but as a canary to avoid breaking database/input parser dependencies.
rasci-c2-active	6-31G* C2 Test RASCI Energy Point, testing two different ways of specifying the active space, either with the ACTIVE keyword, or with RAS1, RAS2, RESTRICTED_DOCC, and RESTRICTED_UOCC
rasci-h2o	RASCI/6-31G** H2O Energy Point
rasci-ne	Ne atom RASCI/cc-pVQZ Example of split-virtual CISD[TQ] from Sherrill and Schaefer, J. Phys. Chem. XXX This uses a "primary" virtual space 3s3p (RAS 2), a "secondary" virtual space 3d4s4p4d4f (RAS 3), and a "tertiary" virtual space consisting of the remaining virtuals. First, an initial CISD computation is run to get the natural orbitals; this allows a meaningful partitioning of the virtual orbitals into groups of different importance. Next, the RASCI is run. The split-virtual CISD[TQ] takes all singles and doubles, and all triples and quadruples with no more than 2 electrons in the secondary virtual subspace (RAS 3). If any electrons are present in the tertiary virtual subspace (RAS 4), then that excitation is only allowed if it is a single or double.
sad1	Test of the superposition of atomic densities (SAD) guess, using a highly distorted water geometry with a cc-pVDZ basis set. This is just a test of the code and the user need only specify guess=sad to the SCF module's (or global) options in order to use a SAD guess. The test is first performed in C2v symmetry, and then in C1.
sapt1	SAPT0 cc-pVDZ computation of the ethene-ethyne interaction energy, using the cc-pVDZ-JKFIT RI basis for SCF and cc-pVDZ-RI for SAPT. Monomer geometries are specified using Cartesian coordinates.

sapt2	SAPT0 aug-cc-pVDZ computation of the benzene-methane interaction energy, using the aug-pVDZ-JKFIT DF basis for SCF, the aug-cc-pVDZ-RI DF basis for SAPT0 induction and dispersion, and the aug-pVDZ-JKFIT DF basis for SAPT0 electrostatics and induction. This example uses frozen core as well as asyncronous I/O while forming the DF integrals and CPHF coefficients.
sapt3	SAPT2+3 aug-cc-pVDZ computation of the water dimer interaction energy, using the aug-cc-pVDZ-JKFIT DF basis for SCF and aug-cc-pVDZ-RI for SAPT.
sapt4	SAPT2+(3) aug-cc-pVDZ computation of the formamide dimer interaction energy, using the aug-cc-pVDZ-JKFIT DF basis for SCF and aug-cc-pVDZ-RI for SAPT. This example uses frozen core as well as MP2 natural orbital approximations.
sapt5	SAPT0 aug-cc-pVTZ computation of the charge transfer energy of the water dimer.
scf-bz2	Benzene Dimer Out-of-Core HF/cc-pVDZ
scf1	RHF cc-pVQZ energy for the BH molecule, with Cartesian input.
scf11-freq-from-energies	Test frequencies by finite differences of energies for planar C4NH4 TS
scf2	RI-SCF cc-pVTZ energy of water, with Z-matrix input and cc-pVTZ-RI auxilliary basis.
scf3	are specified explicitly.
scf4	RHF cc-pVDZ energy for water, automatically scanning the symmetric stretch and bending coordinates using Python's built-in loop mechanisms. The geometry is apecified using a Z-matrix with variables that are updated during the potential energy surface scan, and then the same procedure is performed using polar coordinates, converted to Cartesian coordinates.
scf5	Test of all different algorithms and reference types for SCF, on singlet and triplet O2, using the cc-pVTZ basis set.
tu1-h2o-energy	Sample HF/cc-pVDZ H2O computation
tu2-ch2-energy	Sample UHF/6-31G** CH2 computation

tu3-h2o-opt	Optimize H2O HF/cc-pVDZ
tu4-h2o-freq	Frequencies for H2O HF/cc-pVDZ at optimized geometry
tu5-sapt	Example SAPT computation for ethene*ethine (i.e., ethylene*acetylene), test case 16 from the S22 database
tu6-cp-ne2	Example potential energy surface scan and CP-correction for Ne2
zaptn-nh2	ZAPT(n)/6-31G NH2 Energy Point, with n=2-25

F Basis Set Availability by Element

Table 15: Element availibility for basis sets built into PSI4.

Basis Set	puream	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
Pople																																					
sto-3g	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
3-21g	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
6-31g	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
6-31g_d_	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
6-31g_d_p_	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
6-31gs	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
6-31gss	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
6-31pg	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31pg_d_	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31pg_d_p_	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31pgs	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31pgss	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31ppg	6D/10F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31ppg_d_	6D/10F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31ppg_d_p_	6D/10F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31ppgs	6D/10F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-31ppgss	6D/10F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-311g	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr
6-311g_2d_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-311g_2d_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-311g_2d_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca																
6-311g_2df_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca																
6-311g_2df_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca																
6-311g_2df_2pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca																
6-311g_2df_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca																
6-311g_3df_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
6-311g_3df_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
6-311g_3df_2pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
6-311g_3df_3pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
6-311g_3df_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
6-311g_d_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr
6-311g_d_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr
6-311gs	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr

6-311gss	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Ga	Ge	As	Se	Br	Kr
6-311pg	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pg_2d_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pg_2d_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pg_2d_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pg_2df_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311pg_2df_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311pg_2df_2pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311pg_2df_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311pg_3df_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311pg_3df_2p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311pg_3df_2pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311pg_3df_3pd_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311pg_3df_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311pg_d_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pg_d_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pgs	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311pgss	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg_2d_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg_2d_2p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg_2d_p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg_2df_	5D/7F	H		Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311ppg_2df_2p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311ppg_2df_2pd_	5D/7F	H		Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311ppg_2df_p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne									K	Ca						
6-311ppg_3df_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311ppg_3df_2p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311ppg_3df_2pd_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311ppg_3df_3pd_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311ppg_3df_p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar								
6-311ppg_d_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppg_d_p_	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppgs	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						
6-311ppgss	5D/7F	H		Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca						

Ahlrichs/Karlsruhe

def2-qzvp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-qzvpd	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-qzvpp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-qzvppd	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-sv_p_	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-svp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-svpd	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-tzvp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-tzvpd	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-tzvpp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
def2-tzvppd	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
		H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr

Dunning Dζ

cc-pcvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
cc-pcvdz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
cc-pcvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Ca																
cc-pvdz-canonical	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne																										
cc-pvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar			Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pvdz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
cc-pvdz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr
cc-pvdz-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr
cc-pvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
cc-pwcvdz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
cc-pcv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Ca																
cc-pv_dpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr
cc-pv_dpd_z-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr
cc-pv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
cc-pwcv_dpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																		
aug-cc-pcvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
aug-cc-pcvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
aug-cc-pvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar			Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pvdz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																		
aug-cc-pvdz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr
aug-cc-pvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar													Ga	Ge	As	Se	Br	Kr

aug-cc-pvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pvdzp-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pvdzp-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pvdzp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pwcvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pwcvdz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pcv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
aug-cc-pv_dpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pwcv_dpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
d-aug-cc-pcvdz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pvdz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pwcvdz	5D/7F	H	He			B	C	N	O	F	Ne																								
heavy-aug-cc-pcvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pcvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pvdz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvdz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcvdz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pcv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pv_dpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcv_dpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pcvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
jun-cc-pvdz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvdz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcvdz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pwcvdz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pcv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																

jun-cc-pv_dpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pv_dpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pwcv_dpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
jun-cc-pwcv_dpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	

Dunning Tζ

cc-pcvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pcvtz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pcvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pvtz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pvtz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pvtz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pvtz-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcvtz-dk	5D/7F																				Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
cc-pwcvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pcv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pv_tpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv_tpd_z-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcv_tpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pcvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pcvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pvtz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pvtz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
aug-cc-pvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
aug-cc-pvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcvtz-dk	5D/7F																				Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
aug-cc-pwcvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pcv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pv_tpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr

aug-cc-pv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pwcv_tpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
d-aug-cc-pcvtz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pvtz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne																								
heavy-aug-cc-pcvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pcvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pvtz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvtz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pvtz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcvtz-dk	5D/7F	H	He																	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
heavy-aug-cc-pwcvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pcv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pv_tpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcv_tpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pcvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
jun-cc-pvtz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pcv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
jun-cc-pv_tpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pwcv_tpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
may-cc-pcvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																

may-cc-pvtz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pvtz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pwcvtz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
may-cc-pwcvtz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
may-cc-pcv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
may-cc-pv_tpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pv_tpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pwcv_tpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
may-cc-pwcv_tpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	

Dunning Qζ

cc-pcvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pcvqz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pvqz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pvqz-f12	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pvqz-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcvqz-dk	5D/7F																				Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv_qpd_z-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
aug-cc-pcvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pvqz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
aug-cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
aug-cc-pvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr

aug-cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcvqz-dk	5D/7F																			Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
aug-cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar															Br	
aug-cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
aug-cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
aug-cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
aug-cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar															Br	
d-aug-cc-pcvqz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pvqz	5D/7F	H	He			B	C	N	O	F	Ne																								
d-aug-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne																								
heavy-aug-cc-pcvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pvqz-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvqz-dual	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcvqz-dk	5D/7F	H	He																	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
heavy-aug-cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar															Br	
heavy-aug-cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar															Br	
jun-cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																
jun-cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																
jun-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar															Br	
jun-cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																

jun-cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
jun-cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
jun-cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
may-cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
may-cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
may-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
may-cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
may-cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
may-cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
may-cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
apr-cc-pcvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
apr-cc-pvqz-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pvqz	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pwcvqz-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
apr-cc-pwcvqz	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
apr-cc-pcv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
apr-cc-pv_qpd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pv_qpd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
apr-cc-pwcv_qpd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
apr-cc-pwcv_qpd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																Br	
Dunning 5ζ																																				
cc-pcv5z-dk	5D/7F	H	He			B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pcv5z	5D/7F	H	He			B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pv5z-dk	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pv5z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv5z-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pv5z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr

cc-pwcv5z-dk	5D/7F																				Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
cc-pwcv5z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pcv_5pd_z	5D/7F	H	He			B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca																
cc-pv_5pd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
cc-pv_5pd_z-ri	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar																	
cc-pv_5pd_z	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
cc-pwcv_5pd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
cc-pwcv_5pd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pcv5z-dk	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pcv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pv5z-dk	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pv5z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
aug-cc-pv5z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcv5z-dk	5D/7F																				Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
aug-cc-pwcv5z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pwcv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pcv_5pd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pv_5pd_z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
aug-cc-pv_5pd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pv_5pd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
aug-cc-pwcv_5pd_z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
aug-cc-pwcv_5pd_z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
d-aug-cc-pcv5z	5D/7F	H	He			B	C	N	O	F	Ne																									
d-aug-cc-pv5z	5D/7F	H	He			B	C	N	O	F	Ne																									
d-aug-cc-pwcv5z	5D/7F	H	He			B	C	N	O	F	Ne																									
heavy-aug-cc-pcv5z-dk	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
heavy-aug-cc-pcv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
heavy-aug-cc-pv5z-dk	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv5z-jkfit	5D/7F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar												Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv5z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
heavy-aug-cc-pv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcv5z-dk	5D/7F	H	He																		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn						
heavy-aug-cc-pwcv5z-ri	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	
heavy-aug-cc-pwcv5z	5D/7F	H	He			B	C	N	O	F	Ne			Al	Si	P	S	Cl	Ar																	

heavy-aug-cc-pcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pv_5pd_z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
heavy-aug-cc-pwcv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
heavy-aug-cc-pwcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pv5z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pwcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pv_5pd_z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
jun-cc-pwcv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
jun-cc-pwcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pv5z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
may-cc-pv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
may-cc-pwcv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pwcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pv_5pd_z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
may-cc-pv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
may-cc-pwcv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
may-cc-pwcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
apr-cc-pcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
apr-cc-pv5z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
apr-cc-pv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
apr-cc-pv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar											Ga	Ge	As	Se	Br	Kr
apr-cc-pwcv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																
apr-cc-pwcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar																

apr-cc-pcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
apr-cc-pv_5pd_z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
apr-cc-pv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
apr-cc-pv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
apr-cc-pwcv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
apr-cc-pwcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pv5z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
mar-cc-pv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
mar-cc-pwcv5z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pwcv5z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pv_5pd_z-jkfit	5D/7F	H		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
mar-cc-pv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar	Ga	Ge	As	Se	Br	Kr
mar-cc-pwcv_5pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						
mar-cc-pwcv_5pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar						

Dunning 6ζ

cc-pcv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
cc-pv6z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
cc-pv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
cc-pcv_6pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
cc-pv_6pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
cc-pv_6pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pcv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pv6z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pcv_6pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pv_6pd_z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
aug-cc-pv_6pd_z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
d-aug-cc-pcv6z	5D/7F	H		B	C	N	O								
d-aug-cc-pv6z	5D/7F	H		B	C	N	O								
heavy-aug-cc-pcv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
heavy-aug-cc-pv6z-ri	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
heavy-aug-cc-pv6z	5D/7F	H	He	B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar

heavy-aug-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
heavy-aug-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
heavy-aug-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pcv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pv6z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
jun-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pcv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pv6z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
may-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pcv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pv6z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
apr-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pcv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pv6z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
mar-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pcv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pv6z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pv6z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pcv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pv_6pd_z-ri	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar
feb-cc-pv_6pd_z	5D/7F	H	He		B	C	N	O	F	Ne	Al	Si	P	S	Cl	Ar

Others

dz	5D/7F	H		Li	B	C	N	O	F	Ne	Al	Si	P	S	Cl	
dzp	6D/10F	H		Li	B	C	N	O	F	Ne	Al	Si	P	S	Cl	

dzvp	5D/7F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
g3largexp	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
g4_hf5z	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr
g4_hfqz	6D/10F	H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P	S	Cl	Ar	K	Ca											Ga	Ge	As	Se	Br	Kr
psi3-dzp	6D/10F	H				B	C	N	O	F	Ne			Al	Si	P	S	Cl																			
psi3-tz2p	6D/10F	H				B	C	N	O	F				Al	Si	P	S	Cl																			
psi3-tz2pf	6D/10F	H				B	C	N	O	F				Al	Si	P	S	Cl																			
sadlej-lpol-dl	5D/7F	H					C	N	O	F																											
sadlej-lpol-ds	5D/7F	H					C	N	O	F																											
sadlej-lpol-fl	5D/7F	H					C	N	O	F																											
sadlej-lpol-fs	5D/7F	H					C	N	O	F																											

G Python Driver Functionality

Psithon Documentation

Release 4.01

Psi4 Project

March 04, 2012

CONTENTS

To allow arbitrarily complex computations to be performed, PSI4 is built upon the Python interpreter, with modifications termed Psithon. Chapter 3 of the User's Manual describes the non-standard Python associated with clean molecule, basis, and option specification in the PSI4 input file. This documentation addresses the pure Python side-what functions allow the efficient compiled code to be run, what functions post-process and interact with that output, and how the ordinary (or ambitious) user can extent PSI4's functionality.

Warning: Python naming practices of file_that_includes_function.function_name() are followed below. In psi4 input files, it is only necessary to call the function name alone. That is, use `energy('scf')`, not `driver.energy('scf')`.

Note: The options documented below are placed as arguments in the command that calls the Python function, not in the `set globals` block or with any other `set` command.

Note: Psithon keyword names and values are insensitive to case. The few exceptions are documented for the `database()` function, where case structure must match the database file.

Note: Boolean arguments can be specified by `yes`, `on`, `true`, or `1` for affirmative and `no`, `off`, `false`, or `0` for negative, all insensitive to case.

CHAPTER

ONE

ENERGY

`driver.energy(name, **kwargs)`

Function to compute the single-point electronic energy.

Returns (*float*) Total electronic energy in Hartrees. SAPT returns interaction energy.

Psi variables

CURRENT ENERGY
CURRENT REFERENCE ENERGY
CURRENT CORRELATION ENERGY

name	calls method
scf	Hartree–Fock (HF) or density functional theory (DFT)
mp2	2nd-order Moller-Plesset perturbation theory (MP2)
df-mp2	MP2 with density fitting
dcft	density cumulant functional theory
mcscf	multiconfigurational self consistent field (SCF)
dfcc	coupled cluster with density fitting
mp2c	coupled MP2 (MP2C)
mp2-drpa	random phase approximation?
sapt0	0th-order symmetry adapted perturbation theory (SAPT)
sapt2	2nd-order SAPT, traditional definition
sapt2+	SAPT including all 2nd-order terms
sapt2+(3)	SAPT including perturbative triples
sapt2+3	
sapt0-ct	0th-order SAPT plus charge transfer (CT) calculation
sapt2-ct	SAPT2 plus CT
sapt2+-ct	SAPT2+ plus CT
sapt2+(3)-ct	SAPT2+(3) plus CT
sapt2+3-ct	SAPT2+3 plus CT
cc2	approximate coupled cluster singles and doubles (CC2)
ccsd	coupled cluster singles and doubles (CCSD)
bccd	Brueckner coupled cluster doubles (BCCD)
cc3	approximate coupled cluster singles, doubles, and triples (CC3)
ccsd(t)	CCSD with perturbative triples
bccd(t)	BCCD with perturbative triples
ccenergy	**expert** full control over ccenergy module
mp *n*	*n* th-order Moller–Plesset perturbation theory
zapt *n*	*n* th-order z-averaged perturbation theory (ZAPT)
cisd	configuration interaction (CI) singles and doubles (CISD)

Continued on next page

Table 1.1 – continued from previous page

name	calls method
cisdt	CI singles, doubles, and triples (CISDT)
cisdtq	CI singles, doubles, triples, and quadruples (CISDTQ)
ci *n*	*n* th-order CI
fci	full configuration interaction (FCI)
detci	**expert** full control over detci module
cphf	coupled-perturbed Hartree-Fock?
cpks	coupled-perturbed Kohn-Sham?
cis	CI singles (CIS)
tda	Tamm-Dankoff approximation (TDA)
tdhf	time-dependent HF (TDHF)
tddft	time-dependent DFT (TDDFT)
adc	2nd-order algebraic diagrammatic construction (ADC)
eom-cc2	EOM-CC2
eom-ccsd	equation of motion (EOM) CCSD
eom-cc3	EOM-CC3

name	calls method in Kallay's MRCC program
mrccsd	CC through doubles
mrccsdt	CC through triples
mrccsdtq	CC through quadruples
mrccsdtqp	CC through quintuples
mrccsdtqph	CC through sextuples
mrccsd(t)	CC through doubles with perturbative triples
mrccsdt(q)	CC through triples with perturbative quadruples
mrccsdtq(p)	CC through quadruples with pertubative quintuples
mrccsdtqp(h)	CC through quintuples with pertubative sextuples
mrccsd(t)_l	
mrccsdt(q)_l	
mrccsdtq(p)_l	
mrccsdtqp(h)_l	
mrccsdt-1a	
mrccsdtq-1a	
mrccsdtqp-1a	
mrccsdtqph-1a	
mrccsdt-1b	
mrccsdtq-1b	
mrccsdtqp-1b	
mrccsdtqph-1b	
mrcc2	
mrcc3	
mrcc4	
mrcc5	
mrcc6	
mrccsdt-3	
mrccsdtq-3	
mrccsdtqp-3	
mrccsdtqph-3	

Parameters

- **name** (*string*) – `'scf'` || `'df-mp2'` || `'ci5'` || etc.

  First argument, usually unlabeled. Indicates the computational method to be applied to the system.

- **bypass_scf** (*bool*) – `'on'` || ⇒ `'off'` ⇐

  Indicates whether, for *name* values built atop of scf calculations, the scf step is skipped. Suitable when special steps are taken to get the scf to converge in an explicit preceeding scf step.

Examples

```
>>> # [1] Coupled-cluster singles and doubles calculation with psi code
>>> energy('ccsd')

>>> # [2] Charge-transfer SAPT calculation with scf projection from small into requested basis
>>> energy('sapt0-ct',cast_up=True)

>>> # [3] Arbitrary-order MPn calculation
>>> energy('mp4')
```

CHAPTER

TWO

OPTIMIZE

Note: The derivative level type for `driver.optimize()` and `driver.frequency()` functions can be specified by `energy`, `none`, or `0` for 0th derivative, `gradient`, `first`, or `1` for 1st derivative, and `hessian`, `second`, or `2` for 2nd derivative.

`driver.`**`optimize`**`(`*name*, ***kwargs*`)`

Function to perform a geometry optimization.

Aliases opt()

Returns (*float*) Total electronic energy of optimized structure in Hartrees.

Psi variables

`CURRENT ENERGY`

Note: Analytic gradients area available for all methods in the table below. Optimizations with other methods in the energy table proceed by finite differences.

Caution: Some features are not yet implemented. Buy a developer a coffee.

- Need to check that all methods do return electronic energy. I think gradient got changed at one point.

name	calls method
scf	Hartree–Fock (HF) or density functional theory (DFT)
mp2	2nd-order Moller-Plesset perturbation theory (MP2)
ccsd	coupled cluster singles and doubles (CCSD)
ccsd(t)	CCSD with perturbative triples
eom-ccsd	equation of motion (EOM) CCSD

Parameters

- **name** (*string*) – `'scf'` || `'df-mp2'` || `'ci5'` || etc.

  First argument, usually unlabeled. Indicates the computational method to be applied to the database. May be any valid argument to `driver.energy()`.

- **func** (*function*) – ⇒ `gradient` ⇐ || `energy` || `cbs`

  Indicates the type of calculation to be performed on the molecule. The default dertype accesses‘‘’gradient’‘‘ or `'energy'`, while `'cbs'` performs a multistage finite difference calculation. If a nested series of python functions is intended (see Function Intercalls), use keyword `opt_func` instead of `func`.

- **mode** (*string*) – ⇒ `'continuous'` ⇐ || `'sow'` || `'reap'`

  Indicates whether the calculation required to complete the optimization are to be run in one file (`'continuous'`) or are to be farmed out in an embarrassingly parallel fashion (`'sow'`/`'reap'`). For the latter, run an initial job with `'sow'` and follow instructions in its output file.

- **dertype** (*dertype*) – `'gradient'` || `'energy'`

  Indicates whether analytic (if available) or finite difference optimization is to be performed.

Examples

```
>>> # [1] Analytic scf optimization
>>> optimize('scf')

>>> # [2] Finite difference mp3 optimization
>>> opt('mp3')

>>> # [3] Forced finite difference ccsd optimization
>>> optimize('ccsd', dertype=1)
```

CHAPTER

THREE

RESPONSE

`driver.`**`response`**`(`*name*, ***kwargs*`)`

Function to compute linear response properties.

Returns (*float*) Total electronic energy in Hartrees.

Caution: Some features are not yet implemented. Buy a developer a coffee.

- Check that energy is actually being returned.
- Check if ther're some PSI variables that ought to be set.

name	calls method
cc2	2nd-order approximate CCSD
ccsd	coupled cluster singles and doubles (CCSD)

Parameters **name** (*string*) – `'ccsd'` || etc.

First argument, usually unlabeled. Indicates the computational method to be applied to the system.

Examples

```
>>> # [1] CCSD-LR properties calculation
>>> response('ccsd')
```

CHAPTER

FOUR

FREQUENCY

`driver.`**`frequency`**`(`*name*, ***kwargs*`)`

Function to compute harmonic vibrational frequencies.

Aliases frequencies(), freq()

Returns (*float*) Total electronic energy in Hartrees.

Caution: Some features are not yet implemented. Buy a developer a coffee.

- RAK, why are you adding OPTKING options as GLOBALS? And shouldn't they be Py-side not C-side options?
- Put in a dictionary, so IRREPS can be called by symmetry element or 'all'
- Make frequency look analogous to gradient, especially in matching derivative levels. Make dertype actually a dertype type.

Parameters

- **name** (*string*) – `'scf'` || `'df-mp2'` || `'ci5'` || etc.

  First argument, usually unlabeled. Indicates the computational method to be applied to the system.

- **dertype** (*dertype*) – ⇒ `'hessian'` ⇐ || `'gradient'` || `'energy'`

  Indicates whether analytic (if available- they're not), finite difference of gradients (if available) or finite difference of energies is to be performed.

- **irrep** (*int*) – ⇒ `-1` ⇐ || `1` || etc.

  Indicates which symmetry block of vibrational freqiencies to be computed. 1 represents a_1, requesting only the totally symmetric modes. `-1` indicates a full frequency calculation.

Examples

```
>>> # [1] <example description>
>>> <example python command>

>>> # [2] Frequency calculation for b2 modes through finite difference of gradients
>>> frequencies('scf', dertype=1, irrep=4)
```

CHAPTER

FIVE

COUNTERPOISE CORRECT

`wrappers.cp(name[, func, check_bsse])`

The cp function computes counterpoise-corrected two-body interaction energies for complexes composed of arbitrary numbers of monomers.

Aliases counterpoise_correct(), counterpoise_correction()

Returns (*float*) Counterpoise-corrected interaction energy in kcal/mol

Psi variables

`CP-CORRECTED 2-BODY INTERACTION ENERGY`
`UNCP-CORRECTED 2-BODY INTERACTION ENERGY`

Caution: Some features are not yet implemented. Buy a developer a coffee.
- No values of func besides energy have been tested.
- Table print-out needs improving. Add some PSI variables.

Parameters

- **name** (*string*) – `'scf'` || `'ccsd(t)'` || etc.

  First argument, usually unlabeled. Indicates the computational method to be applied to the molecule. May be any valid argument to `driver.energy()`; however, SAPT is not appropriate.

- **func** (*function*) – ⇒ `energy` ⇐ || `optimize` || `cbs`

  Indicates the type of calculation to be performed on the molecule and each of its monomers. The default performs a single-point `energy('name')`, while `optimize` perfoms a geometry optimization on each system, and `cbs` performs a compound single-point energy. If a nested series of python functions is intended (see Function Intercalls), use keyword `cp_func` instead of `func`.

- **check_bsse** (*bool*) – `'on'` || ⇒ `'off'` ⇐

  Indicates whether to additionally compute un-counterpoise corrected monomers and thus obtain an estimate for the basis set superposition error.

Examples

```
>>> # [1] counterpoise-corrected mp2 interaction energy
>>> cp('dfmp2')
```

CHAPTER

SIX

DATABASE

`wrappers.`**`database`**`(`*name*, *db_name*[, *func*, *mode*, *cp*, *rlxd*, *symm*, *zpe*, *benchmark*, *tabulate*, *subset*]`)`

Function to access the molecule objects and reference energies of popular chemical databases.

Aliases db()

Returns (*float*) Mean absolute deviation of the database in kcal/mol

Psi variables

`db_name DATABASE MEAN SIGNED DEVIATION`
`db_name DATABASE MEAN ABSOLUTE DEVIATION`
`db_name DATABASE ROOT-MEAN-SQUARE DEVIATION`

Note: It is very easy to make a database from a collection of xyz files using the script `$PSIDATADIR/databases/ixyz2database.pl`. See **'Creating a New Database'_** for details.

Caution: Some features are not yet implemented. Buy a developer some coffee.

- In sow/reap mode, use only global options (e.g., the local option set by `set scf scf_type df` will not be respected).

Parameters

- **name** (*string*) – `'scf'` || `'sapt0'` || `'ccsd(t)'` || etc.

  First argument, usually unlabeled. Indicates the computational method to be applied to the database. May be any valid argument to `driver.energy()`.

- **db_name** (*string*) – `'BASIC'` || `'S22'` || `'HTBH'` || etc.

  Second argument, usually unlabeled. Indicates the requested database name, matching the name of a python file in `psi4/lib/databases`. Consult that directory for available databases and literature citations.

- **func** (*function*) – ⇒ `energy` ⇐ || `optimize` || `cbs`

  Indicates the type of calculation to be performed on each database member. The default performs a single-point `energy('name')`, while `optimize` perfoms a geometry optimization on each reagent, and `cbs` performs a compound single-point energy. If a nested series of python functions is intended (see Function Intercalls), use keyword `db_func` instead of `func`.

- **mode** (*string*) – ⇒ `'continuous'` ⇐ || `'sow'` || `'reap'`

  Indicates whether the calculations required to complete the database are to be run in one file (`'continuous'`) or are to be farmed out in an embarrassingly parallel fashion ('sow'/'reap'). For the latter, run an initial job with 'sow' and follow instructions in its output file.

- **cp** (*bool*) – 'on' || ⇒ 'off' ⇐

  Indicates whether counterpoise correction is employed in computing interaction energies. Use this option and NOT the `wrappers.cp()` function for BSSE correction in database(). Option available (See Available Databases) only for databases of bimolecular complexes.

- **rlxd** (*bool*) – 'on' || ⇒ 'off' ⇐

  Indicates whether correction for deformation energy is employed in computing interaction energies. Option available (See Available Databases) only for databases of bimolecular complexes with non-frozen monomers, e.g., HBC6.

- **symm** (*bool*) – ⇒ 'on' ⇐ || 'off'

  Indicates whether the native symmetry of the database reagents is employed ('on') or whether it is forced to C_1 symmetry ('off'). Some computational methods (e.g., SAPT) require no symmetry, and this will be set by database().

- **zpe** (*bool*) – 'on' || ⇒ 'off' ⇐

  Indicates whether zero-point-energy corrections are appended to single-point energy values. Option valid only for certain thermochemical databases. Disabled until Hessians ready.

- **benchmark** (*string*) – ⇒ 'default' ⇐ || 'S22A' || etc.

  Indicates whether a non-default set of reference energies, if available (See Available Databases), are employed for the calculation of error statistics.

- **tabulate** (*array of strings*) – ⇒ [] ⇐ || ['scf total energy', 'natom'] || etc.

  Indicates whether to form tables of variables other than the primary requested energy. Available for any PSI variable.

- **subset** (*string or array of strings*) – Indicates a subset of the full database to run. This is a very flexible option and can be used in three distinct ways, outlined below. Note that two take a string and the last takes an array. See Available Databases for available values.

  - **'small' || 'large' || 'equilibrium'** Calls predefined subsets of the requested database, either 'small', a few of the smallest database members, 'large', the largest of the database members, or 'equilibrium', the equilibrium geometries for a database composed of dissociation curves.

  - **'BzBz_S' || 'FaOOFaON' || 'ArNe' || etc.** For databases composed of dissociation curves, individual curves can be called by name. Consult the database python files for available molecular systems. The choices for this keyword are case sensitive and must match the database python file

  - **[1,2,5] || ['1','2','5'] || ['BzMe-3.5', 'MeMe-5.0'] || etc.** Specify a list of database members to run. Consult the database python files for available molecular systems. The choices for this keyword are case sensitive and must match the database python file

Examples

```
>>> # [1] Two-stage SCF calculation on short, equilibrium, and long helium dimer
>>> db('scf','RGC10',cast_up='sto-3g',subset=['HeHe-0.85','HeHe-1.0','HeHe-1.5'], tabulate=['scf

>>> # [2] Counterpoise-corrected interaction energies for three complexes in S22
>>> #     Error statistics computed wrt an old benchmark, S22A
>>> database('dfmp2','S22',cp=1,subset=[16,17,8],benchmark='S22A')
>>> # [3] SAPT0 on the neon dimer dissociation curve
>>> db('sapt0',subset='NeNe',cp=0,symm=0,db_name='RGC10')

>>> # [4] Optimize system 1 in database S22, producing tables of scf and mp2 energy
>>> db('mp2','S22',db_func=optimize,subset=[1], tabulate=['mp2 total energy','current energy'])

>>> # [5] CCSD on the smallest systems of HTBH, a hydrogen-transfer database
>>> database('ccsd','HTBH',subset='small', tabulate=['ccsd total energy', 'mp2 total energy'])
```

6.1 Output

At the beginning of a database job is printed a listing of the individual system calculations which will be performed. The output snippet below is from the example job [1] above. It shows each reagent required for the subset of database reactions requested. Note that this is an un-counterpoise-corrected example, and the wrapper is smart enough to compute only once the monomer whose energy will be subtracted from each of the three dimers.

```
RGC1-HeHe-0.85-dimer
RGC1-He-mono-unCP
RGC1-HeHe-1.0-dimer
RGC1-HeHe-1.5-dimer
```

At the end of the job, the Requested Energy table is printed that gives the total energies for the requested model chemistry for each reagent and each reaction, as well as the stoichoimetric weights by which the reagent energies are transfromed into the reaction energy. In this case, the dimer is +1 and the monomer is -2, indicating the the interaction energy is computed from dimer less first monomer less second (identical) monomer. Error statistics are computed with respect to the reference energies stored in the database. One of these, the mean absolute deviation, is returned by the wrapper as an ordinary Python variable. (For databases without a stored reference energy, e.g., BASIC, large and meaningless numbers are printed for error.) The other two tables tabulate the PSI variables requested through keyword `tabulate`, in this case the total SCF energy and the number of atoms in each reagent.

```
==> Scf Total Energy <==

-----------------------------------------------------------------------------------
         Reaction          Reaction Value                Reagent 1        Reagent 2
                                                          Value Wt         Value Wt
-----------------------------------------------------------------------------------
   RGC1-HeHe-0.85             0.00011520           -5.71020576  1  -2.85516048 -2
    RGC1-HeHe-1.0             0.00000153           -5.71031943  1  -2.85516048 -2
    RGC1-HeHe-1.5            -0.00000000           -5.71032096  1  -2.85516048 -2
-----------------------------------------------------------------------------------

==> Natom <==

-----------------------------------------------------------------------------------
         Reaction          Reaction Value                Reagent 1        Reagent 2
                                                          Value Wt         Value Wt
-----------------------------------------------------------------------------------
   RGC1-HeHe-0.85             0.00000000            2.00000000  1   1.00000000 -2
    RGC1-HeHe-1.0             0.00000000            2.00000000  1   1.00000000 -2
    RGC1-HeHe-1.5             0.00000000            2.00000000  1   1.00000000 -2
-----------------------------------------------------------------------------------

==> Requested Energy <==

-----------------------------------------------------------------------------------
```

```
        Reaction     Reaction Energy      Error       Reagent 1       Reagent 2
                        Ref     Calc [kcal/mol]          [H] Wt          [H] Wt
-----------------------------------------------------------------------------------
   RGC1-HeHe-0.85     0.0376   0.0723      0.0347  -5.71020576  1  -2.85516048 -2
    RGC1-HeHe-1.0    -0.0219   0.0010      0.0228  -5.71031943  1  -2.85516048 -2
    RGC1-HeHe-1.5    -0.0029  -0.0000      0.0029  -5.71032096  1  -2.85516048 -2
-----------------------------------------------------------------------------------
      Minimal Dev                          0.0029
      Maximal Dev                          0.0347
  Mean Signed Dev                          0.0201
Mean Absolute Dev                          0.0201
          RMS Dev                          0.0240
-----------------------------------------------------------------------------------
```

6.2 Available Databases

Below are documented for particular databases the availibility of the generic database function options **cp**, **rlxd**, **benchmark**, and the string options for **subset**. The full reagent member list, which can also be used in conjunction with **subset**, is not included here for consideration of space and may be found in the database file. The database Python files are very readable and should be consulted for more particular questions.

ACENES

Database of Ed and Rob's favorite linear acene dimers.
Geometries from nowhere special, and reference energies undefined.

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`
  - `'FIRST3'` benzene, napthalene, and anthracene dimers
  - `'FIRST5'` benzene - pentacene dimers
  - `'FIRST10'` benzene - decacene dimers

BAKERJCC93

Database of molecules that are challenging to optimize.
Geometries from Baker J. Comput. Chem. 14 1085 (1993), as reported in Bakken and Helgaker, J. Chem. Phys. 117, 9160 (2002), with a few further corrections.
No reference energies defined.

- **cp** `'off'`

- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`

BASIC

Database of simple molecules, mostly for testing.
Geometries from nowhere special, and no reference energies defined.

- **cp** `'off'`
- **rlxd** `'off'`
- **subset** `['h2o','nh3','ch4']`

BBI

Database (Merz) of protein backbone-backbone interactions.
Geometries from Kenneth Merz Group, Univ. of Florida.
Reference interaction energies from Sherrill group, Georgia Tech.

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`

CFLOW

Database of extended conjugated bimolecular systems.
Geometries and Reference interaction energies from the following articles:

Polyene geometries from Marshall et al. JCTC 6 3681 (2010).
Polyene reference interaction energies from Sherrill group by ccsd(t**)-f12b/heavy-aug-cc-pvdz.
Acene geometries (except benzene) from Sherrill group by df-mp2/cc-pvtz c.2011.
Benzene geometry from NBC10 database and citations therein.
Acene reference interaction energies (incl. benzene dimer) from Sherrill group by ccsd(t**)-f12b/aug-cc-pvdz.
Buckybowl (Pulay-labeled) geometries from Sherrill group by PBE1PBE/6-31G*, following Pulay's instructions in Janowski et al. CPL 512 155 (2011).
Buckybowl (Pulay-labeled) reference interaction energies from Janowski et al. CPL 512 155 (2011).
Buckyware (Grimme-labeled) geometries from Grimme PCCP 12 7091 (2010).
Buckyware (Grimme-labeled) reference interaction energies from Grimme PCCP 12 7091 (2010) by B97-D2/TZVP.
Collection into CFLOW by Parrish et al. XXX XXX XXXXXX (2012).

- **cp** `'off'` || `'on'`

- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`
  - `'equilibrium'`
  - `'Polyenes'` equilibrium for linear polyene dimers for 2 through 16 monomer carbons
  - `'cBzBz'` 5-point dissociation curve for benzene dimer
  - `'c2BzBz'` 5-point dissociation curve for napthalene-benzene complex
  - `'c2Bz2Bz'` 5-point dissociation curve for napthalene dimer
  - `'c3Bz2Bz'` 5-point dissociation curve for anthracene-napthalene complex
  - `'c3Bz3Bz'` 5-point dissociation curve for anthracene dimer
  - `'c4Bz3Bz'` 5-point dissociation curve for tetracene-anthracene complex
  - `'Arenes'` equilibrium for benzene dimer through tetracene-anthracene complex linear arenes
  - `'cArenes'` 5-point curves around benzene dimer through tetracene-anthracene complex linear arenes
  - `'cPulay'` 4-point dissociation curve for bowl-in-bowl corannulene dimer
  - `'Pulay'` Pulay bowl-in-bowl corannulene dimer dissociation curve and extra point
  - `'Grimme60'` Grimme corannulene dimer, C60 @ buckybowl, and C60 @ buckycatcher
  - `'Grimme70'` Grimme C70 @ buckycatcher at three orientations
  - `'Paper'` linear polyene dimers, equilibrium arene complexes, Pulay corannulene dimer curve, and Grimme corannulene dimer and C60 complexes
  - `'cPaper'` linear polyene dimers, arene complex curves, Pulay corannulene dimer curve, and Grimme corannulene dimer and C60 complexes

---

CORE

Database of Pulay corannulene structures. Subsumed into CFLOW.

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`

G2

Database of thermodynamic reactions.
WIP

- **cp** `'off'`
- **rlxd** `'off'`

HBC6

Database (Sherrill) of interaction energies for dissociation curves of doubly hydrogen-bonded bimolecular complexes.
Geometries from Thanthiriwatte et al. JCTC 7 88 (2011).
Reference interaction energies from Marshall et al. JCP 135 194102 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'` || `'on'`
- **benchmark**
  - `'HBC60'` Thanthiriwatte et al. JCTC 7 88 (2011).
  - ⇒ `'HBC6A'` ⇐ Marshall et al. JCP 135 194102 (2011).
  - `'HBC6ARLX'` Sherrill group, unpublished.
- **subset**
  - `'small'`
  - `'large'`
  - `'equilibrium'`

HSG

Database (Merz) of interaction energies for bimolecular complexes from protein-indinavir reaction site.
Geometries from Faver et al. JCTC 7 790 (2011).
Reference interaction energies from Marshall et al. JCP 135 194102 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **benchmark**
  - `'HSG0'` Faver et al. JCTC 7 790 (2011).
  - ⇒ `'HSGA'` ⇐ Marshall et al. JCP 135 194102 (2011).
- **subset**
  - `'small'`
  - `'large'`

HTBH

Database (Truhlar) of hydrogen-transfer barrier height reactions.
Geometries from Truhlar and coworkers at site
http://t1.chem.umn.edu/misc/database_group/database_therm_bh/raw_geom.cgi .
Reference energies from Zhao et al. JPCA, 109 2012-2018 (2005) doi: 10.1021/jp045141s [in supporting information].

- **cp** `'off'`

- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`

JSCH

Database (Hobza) of interaction energies for nucelobase pairs.
Geometries and reference interaction energies from Jurecka et al. PCCP 8 1985 (2006).
Corrections implemented from footnote 92 of Burns et al., JCP 134 084107 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`

NBC10

Database (Sherrill) of interaction energies for dissociation curves of dispersion-bound bimolecular complexes.
Geometries and Reference interaction energies from the following articles:

> Benzene Dimers from Sherrill et al. JPCA 113 10146 (2009).
> Benzene-Hydrogen Sulfide from Sherrill et al. JPCA 113 10146 (2009).
> Benzene-Methane from Sherrill et al. JPCA 113 10146 (2009).
> Methane Dimer from Takatani et al. PCCP 9 6106 (2007).
> Pyridine Dimers from Hohenstein et al. JPCA 113 878 (2009).
> Collection into NBC10 from Burns et al. JCP 134 084107 (2011).
> Reference from Marshall et al. JCP 135 194102 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **benchmark**
  - `'NBC100'` Burns et al. JCP 134 084107 (2011).
  - $\Rightarrow$ `'NBC10A'` $\Leftarrow$ Marshall et al. JCP 135 194102 (2011).
- **subset**
  - `'small'`
  - `'large'`
  - `'equilibrium'`
  - `'BzBz_S'` dissociation curve for benzene dimer, sandwich

- 'BzBz_T' dissociation curve for benzene dimer, t-shaped
- 'BzBz_PD34' dissociation curve for benzene dimer, parallel displaced by 3.4A
- 'BzH2S' dissociation curve for benzene-H2S
- 'BzMe' dissociation curve for benzene-methane
- 'MeMe' dissociation curve for methane dimer
- 'PyPy_S2' dissociation curve for pyridine dimer, sandwich
- 'PyPy_T3' dissociation curve for pyridine dimer, t-shaped
- 'BzBz_PD32' dissociation curve for benzene dimer, parallel displaced by 3.2A
- 'BzBz_PD36' dissociation curve for benzene dimer, parallel displaced by 3.6A

NHTBH

Database (Truhlar) of non-hydrogen-transfer barrier height reactions.
Geometries and Reaction energies from Truhlar and coworkers at site
http://t1.chem.umn.edu/misc/database_group/database_therm_bh/non_H.htm.

- **cp** 'off'
- **rlxd** 'off'
- **subset**
  - 'small'
  - 'large'

RGC10

Database (Sherrill) of interaction energies for dissociation curves of rare-gas biatomic complexes.
Geometries and reference interaction energies from Tang et al. JCP 118 4976 (2003).

- **cp** 'off' || 'on'
- **rlxd** 'off'
- **subset**
  - 'small'
  - 'large'
  - 'equilibrium'
  - 'HeHe' 18-point dissociation curve for helium dimer
  - 'HeNe' 18-point dissociation curve for helium-neon complex
  - 'HeAr' 18-point dissociation curve for helium-argon complex
  - 'HeKr' 18-point dissociation curve for helium-krypton complex

- 'NeNe' 18-point dissociation curve for neon dimer
- 'NeAr' 18-point dissociation curve for neon-argon complex
- 'NeKr' 18-point dissociation curve for neon-krypton complex
- 'ArAr' 18-point dissociation curve for argon dimer
- 'ArKr' 18-point dissociation curve for argon-krypton complex
- 'KrKr' 18-point dissociation curve for krypton dimer

S22

Database (Hobza) of interaction energies for bimolecular complexes.
Geometries from Jurecka et al. PCCP 8 1985 (2006).
Reference interaction energies from Marshall et al. JCP 135 194102 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **benchmark**
  - `'S220'` Jurecka et al. PCCP 8 1985 (2006).
  - `'S22A'` Takatani et al. JCP 132 144104 (2010).
  - ⇒ `'S22B'` ⇐ Marshall et al. JCP 135 194102 (2011).
- **subset**
  - `'small'` water dimer, methane dimer, ethene-ethine
  - `'large'` adenine-thymine
  - `'HB'` hydrogen-bonded systems
  - `'MX'` mixed-influence systems
  - `'DD'` dispersion-dominated systems

S22by5

Database (Hobza) of interaction energies for dissociation curves of bimolecular complexes.
Geometries and reference interaction energies from Grafova et al. JCTC 6 2365 (2010).
Note that the S22by5-N-1.0 members are essentially the same geometries as S22-N (there's trivial round-off error) but the reference interaction energies for S22by5 are of lower quality than those of S22.

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`

- `'equilibrium'`
- `'mol1'` five-point (0.9, 1.0, 1.2, 1.5, 2.0) $\times R_{eq}$ dissociation curve for molecule 1
- ...
- `'mol22'` five-point (0.9, 1.0, 1.2, 1.5, 2.0) $\times R_{eq}$ dissociation curve for molecule 22

S66

Database (Hobza) of interaction energies for bimolecular complexes.
Geometries and reference energies from Rezac et al. JCTC 7 2427 (2011).

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`
- **subset**
  - `'small'`
  - `'large'`
  - `'HB'` hydrogen-bonded systems
  - `'MX'` mixed-influence systems
  - `'DD'` dispersion-dominated systems

SSI

Database (Merz) of interaction energies for protein sidechain-sidechain interactions.
Geometries from Kenneth Merz Group, Univ. of Florida.
Reference interaction energies from <Reference>.

- **cp** `'off'` || `'on'`
- **rlxd** `'off'`

CHAPTER

SEVEN

COMPLETE BASIS SET

wrappers.**complete_basis_set**(*name*[, *scf_basis*, *scf_scheme*, *corl_wfn*, *corl_basis*, *corl_scheme*, *delta_wfn*, *delta_wfn_lesser*, *delta_basis*, *delta_scheme*, *delta2_wfn*, *delta2_wfn_lesser*, *delta2_basis*, *delta2_scheme*])

Function to define a multistage energy method from combinations of basis set extrapolations and delta corrections and condense the components into a minimum number of calculations.

Aliases cbs()

Returns (*float*) – Total electronic energy in Hartrees

Psi variables

CBS TOTAL ENERGY
CBS REFERENCE ENERGY
CBS CORRELATION ENERGY
CURRENT ENERGY
CURRENT REFERENCE ENERGY
CURRENT CORRELATION ENERGY

Caution: Some features are not yet implemented. Buy a developer a coffee.

- Methods beyond basic scf, mp2, ccsd, ccsd(t) not yet hooked in through PSI variables, df-mp2 in particular.
- No scheme defaults for given basis zeta number, so scheme must be specified explicitly.
- No way to tell function to boost fitting basis size for all calculations.
- No way to extrapolate def2 family basis sets
- Need to add more extrapolation schemes

As represented in the equation below, a CBS energy method is defined in four sequential stages (scf, corl, delta, delta2) covering treatment of the reference total energy, the correlation energy, a delta correction to the correlation energy, and a second delta correction. Each is activated by its stage_wfn keyword and is only allowed if all preceding stages are active.

$$E_{total}^{\text{CBS}} = \mathcal{F}_{\textbf{scf\_scheme}}\left(E_{total,\ \text{SCF}}^{\textbf{scf\_basis}}\right) + \mathcal{F}_{\textbf{corl\_scheme}}\left(E_{corl,\ \textbf{corl\_wfn}}^{\textbf{corl\_basis}}\right) + \delta_{\textbf{delta\_wfn\_lesser}}^{\textbf{delta\_wfn}} + \delta_{\textbf{delta2\_wfn\_lesser}}^{\textbf{delta2\_wfn}}$$

Here, $\mathcal{F}$ is an energy or energy extrapolation scheme, and the following also hold.

$$\delta_{\textbf{delta\_wfn\_lesser}}^{\textbf{delta\_wfn}} = \mathcal{F}_{\textbf{delta\_scheme}}\left(E_{corl,\ \textbf{delta\_wfn}}^{\textbf{delta\_basis}}\right) - \mathcal{F}_{\textbf{delta\_scheme}}\left(E_{corl,\ \textbf{delta\_wfn\_lesser}}^{\textbf{delta\_basis}}\right)$$

$$\delta_{\textbf{delta2\_wfn\_lesser}}^{\textbf{delta2\_wfn}} = \mathcal{F}_{\textbf{delta2\_scheme}}\left(E_{corl,\ \textbf{delta2\_wfn}}^{\textbf{delta2\_basis}}\right) - \mathcal{F}_{\textbf{delta2\_scheme}}\left(E_{corl,\ \textbf{delta2\_wfn\_lesser}}^{\textbf{delta2\_basis}}\right)$$

A translation of this ungainly equation to example [5] below is as follows. In words, this is a double- and triple-zeta 2-point Helgaker-extrapolated CCSD(T) coupled-cluster correlation correction appended to a triple- and quadruple-zeta 2-point Helgaker-extrapolated MP2 correlation energy appended to a SCF/aug-cc-pVQZ reference energy.

$$E_{total}^{\text{CBS}} = \mathcal{F}_{\text{highest\_1}}\left(E_{total,\ \text{SCF}}^{\text{aug-cc-pVQZ}}\right) + \mathcal{F}_{\text{corl\_xtpl\_helgaker\_2}}\left(E_{corl,\ \text{MP2}}^{\text{aug-cc-pV[TQ]Z}}\right) + \delta_{\text{MP2}}^{\text{CCSD(T)}}$$

$$\delta_{\text{MP2}}^{\text{CCSD(T)}} = \mathcal{F}_{\text{corl\_xtpl\_helgaker\_2}}\left(E_{corl,\ \text{CCSD(T)}}^{\text{aug-cc-pV[DT]Z}}\right) - \mathcal{F}_{\text{corl\_xtpl\_helgaker\_2}}\left(E_{corl,\ \text{MP2}}^{\text{aug-cc-pV[DT]Z}}\right)$$

•**Energy Methods** The presence of a stage_wfn keyword is the indicator to incorporate (and check for stage_basis and stage_scheme keywords) and compute that stage in defining the CBS energy.

The cbs() function requires, at a minimum, `name='scf'` and `scf_basis` keywords to be specified for reference-step only jobs and `name` and `corl_basis` keywords for correlated jobs.

Parameters

- **name** (*string*) – `'scf'` || `'ccsd'` || etc.

  First argument, usually unlabeled. Indicates the computational method for the correlation energy, unless only reference step to be performed, in which case should be `'scf'`. Overruled if stage_wfn keywords supplied.

- **corl_wfn** (*string*) – `'mp2'` || `'ccsd(t)'` || etc.

  Indicates the energy method for which the correlation energy is to be obtained. Can also be specified with `name` or as the unlabeled first argument to the function.

- **delta_wfn** (*string*) – `'ccsd'` || `'ccsd(t)'` || etc.

  Indicates the (superior) energy method for which a delta correction to the correlation energy is to be obtained.

- **delta_wfn_lesser** (*string*) – $\Rightarrow$ `'mp2'` $\Leftarrow$ || `'ccsd'` || etc.

  Indicates the inferior energy method for which a delta correction to the correlation energy is to be obtained.

- **delta2_wfn** (*string*) – `'ccsd'` || `'ccsd(t)'` || etc.

  Indicates the (superior) energy method for which a second delta correction to the correlation energy is to be obtained.

- **delta2_wfn_lesser** (*string*) – $\Rightarrow$ `'mp2'` $\Leftarrow$ || `'ccsd(t)'` || etc.

  Indicates the inferior energy method for which a second delta correction to the correlation energy is to be obtained.

•**Basis Sets** Currently, the basis set set through `set` commands have no influence on a cbs calculation.

Parameters

- **scf_basis** (*string*) – ⇒ `corl_basis` ⇐ || `'cc-pV[TQ]Z'` || `'jun-cc-pv[tq5]z'` || `'6-31G*'` || etc.

  Indicates the sequence of basis sets employed for the reference energy. If any correlation method is specified, `scf_basis` can default to `corl_basis`.

- **corl_basis** (*string*) – `'cc-pV[TQ]Z'` || `'jun-cc-pv[tq5]z'` || `'6-31G*'` || etc.

  Indicates the sequence of basis sets employed for the correlation energy.

- **delta_basis** (*string*) – `'cc-pV[TQ]Z'` || `'jun-cc-pv[tq5]z'` || `'6-31G*'` || etc.

  Indicates the sequence of basis sets employed for the delta correction to the correlation energy.

- **delta2_basis** (*string*) – `'cc-pV[TQ]Z'` || `'jun-cc-pv[tq5]z'` || `'6-31G*'` || etc.

  Indicates the sequence of basis sets employed for the second delta correction to the correlation energy.

•**Schemes** Transformations of the energy through basis set extrapolation for each stage of the CBS definition. A complaint is generated if number of basis sets in stage_basis does not exactly satisfy requirements of stage_scheme. An exception is the default, `'highest_1'`, which uses the best basis set available. See Extrapolation Schemes for all available schemes.

Parameters

- **scf_scheme** (*function*) – ⇒ `highest_1` ⇐ || `scf_xtpl_helgaker_3` || etc.

  Indicates the basis set extrapolation scheme to be applied to the reference energy.

- **corl_scheme** (*function*) – ⇒ `highest_1` ⇐ || `corl_xtpl_helgaker_2` || etc.

  Indicates the basis set extrapolation scheme to be applied to the correlation energy.

- **delta_scheme** (*function*) – ⇒ `highest_1` ⇐ || `corl_xtpl_helgaker_2` || etc.

  Indicates the basis set extrapolation scheme to be applied to the delta correction to the correlation energy.

- **delta2_scheme** (*function*) – ⇒ `highest_1` ⇐ || `corl_xtpl_helgaker_2` || etc.

  Indicates the basis set extrapolation scheme to be applied to the second delta correction to the correlation energy.

Examples

```
>>> # [1] replicates with cbs() the simple model chemistry scf/cc-pVDZ: set basis cc-pVDZ energy
>>> cbs('scf', scf_basis='cc-pVDZ')

>>> # [2] replicates with cbs() the simple model chemistry mp2/jun-cc-pVDZ: set basis jun-cc-pVD
>>> cbs('mp2', corl_basis='jun-cc-pVDZ')

>>> # [3] DTQ-zeta extrapolated scf reference energy
>>> cbs('scf', scf_basis='cc-pV[DTQ]Z', scf_scheme=scf_xtpl_helgaker_3)

>>> # [4] DT-zeta extrapolated mp2 correlation energy atop a T-zeta reference
>>> cbs('mp2', corl_basis='cc-pv[dt]z', corl_scheme=corl_xtpl_helgaker_2)

>>> # [5] a DT-zeta extrapolated coupled-cluster correction atop a TQ-zeta extrapolated mp2 corr
>>> cbs('mp2', corl_basis='aug-cc-pv[tq]z', corl_scheme=corl_xtpl_helgaker_2, delta_wfn='ccsd(t)
>>> # [6] a D-zeta ccsd(t) correction atop a DT-zeta extrapolated ccsd cluster correction atop a
>>> cbs('mp2', corl_basis='aug-cc-pv[tq]z', corl_scheme=corl_xtpl_helgaker_2, delta_wfn='ccsd',

>>> # [7] cbs() coupled with database()
>>> database('mp2', 'BASIC', subset=['h2o','nh3'], symm='on', func=cbs, corl_basis='cc-pV[tq]z',
```

7.1 Output

At the beginning of a cbs() job is printed a listing of the individual energy calculations which will be performed. The output snippet below is from the example job [2] above. It shows first each model chemistry needed to compute the aggregate model chemistry requested through cbs(). Then, since, for example, an `energy('ccsd(t)')` yields CCSD(T), CCSD, MP2, and SCF energy values, the wrapper condenses this task list into the second list of minimum number of calculations which will actually be run.

```
Naive listing of computations required.
        scf / aug-cc-pvqz              for  SCF TOTAL ENERGY
        mp2 / aug-cc-pvtz              for  MP2 CORRELATION ENERGY
        mp2 / aug-cc-pvqz              for  MP2 CORRELATION ENERGY
    ccsd(t) / aug-cc-pvdz              for  CCSD(T) CORRELATION ENERGY
    ccsd(t) / aug-cc-pvtz              for  CCSD(T) CORRELATION ENERGY
        mp2 / aug-cc-pvdz              for  MP2 CORRELATION ENERGY
        mp2 / aug-cc-pvtz              for  MP2 CORRELATION ENERGY

Enlightened listing of computations required.
        mp2 / aug-cc-pvqz              for  MP2 CORRELATION ENERGY
    ccsd(t) / aug-cc-pvdz              for  CCSD(T) CORRELATION ENERGY
    ccsd(t) / aug-cc-pvtz              for  CCSD(T) CORRELATION ENERGY
```

At the end of a cbs() job is printed a summary section like the one below. First, in the components section, are listed the results for each model chemistry available, whether required for the cbs job (*) or not. Next, in the stages section, are listed the results for each extrapolation. The energies of this section must be dotted with the weightings in column Wt to get the total cbs energy. Finally, in the CBS section, are listed the results for each stage of the cbs procedure. The stage energies of this section sum outright to the total cbs energy.

```
==> Components <==

---------------------------------------------------------------------------------
               Method / Basis            Rqd   Energy [H]   Variable
---------------------------------------------------------------------------------
                  scf / aug-cc-pvqz        *  -1.11916375   SCF TOTAL ENERGY
                  mp2 / aug-cc-pvqz        *  -0.03407997   MP2 CORRELATION ENERGY
                  scf / aug-cc-pvdz           -1.11662884   SCF TOTAL ENERGY
                  mp2 / aug-cc-pvdz        *  -0.02881480   MP2 CORRELATION ENERGY
              ccsd(t) / aug-cc-pvdz        *  -0.03893812   CCSD(T) CORRELATION ENERGY
                 ccsd / aug-cc-pvdz           -0.03893812   CCSD CORRELATION ENERGY
                  scf / aug-cc-pvtz           -1.11881134   SCF TOTAL ENERGY
                  mp2 / aug-cc-pvtz        *  -0.03288936   MP2 CORRELATION ENERGY
              ccsd(t) / aug-cc-pvtz        *  -0.04201004   CCSD(T) CORRELATION ENERGY
                 ccsd / aug-cc-pvtz           -0.04201004   CCSD CORRELATION ENERGY
---------------------------------------------------------------------------------

==> Stages <==

---------------------------------------------------------------------------------
 Stage         Method / Basis             Wt   Energy [H]   Scheme
---------------------------------------------------------------------------------
```

```
   scf             scf / aug-cc-pvqz        1  -1.11916375   highest_1
  corl             mp2 / aug-cc-pv[tq]z     1  -0.03494879   corl_xtpl_helgaker_2
 delta         ccsd(t) / aug-cc-pv[dt]z     1  -0.04330347   corl_xtpl_helgaker_2
 delta             mp2 / aug-cc-pv[dt]z    -1  -0.03460497   corl_xtpl_helgaker_2
---------------------------------------------------------------------------------

==> CBS <==

---------------------------------------------------------------------------------
 Stage         Method / Basis                  Energy [H]   Scheme
---------------------------------------------------------------------------------
   scf             scf / aug-cc-pvqz           -1.11916375   highest_1
  corl             mp2 / aug-cc-pv[tq]z        -0.03494879   corl_xtpl_helgaker_2
 delta  ccsd(t) - mp2 / aug-cc-pv[dt]z         -0.00869851   corl_xtpl_helgaker_2
 total             CBS                         -1.16281105
---------------------------------------------------------------------------------
```

7.2 Extrapolation Schemes

`wrappers.`**`highest_1`**`(`*`**largs`*`)`

Scheme for total or correlation energies with a single basis or the highest zeta-level among an array of bases. Used by `wrappers.complete_basis_set()`.

$$E_{total}^X = E_{total}^X$$

`wrappers.`**`scf_xtpl_helgaker_2`**`(`*`**largs`*`)`

Extrapolation scheme for reference energies with two adjacent zeta-level bases. Used by `wrappers.complete_basis_set()`.

$$E_{total}^X = E_{total}^\infty + \beta e^{-\alpha X}, \alpha = 1.63$$

`wrappers.`**`scf_xtpl_helgaker_3`**`(`*`**largs`*`)`

Extrapolation scheme for reference energies with three adjacent zeta-level bases. Used by `wrappers.complete_basis_set()`.

$$E_{total}^X = E_{total}^\infty + \beta e^{-\alpha X}$$

`wrappers.`**`corl_xtpl_helgaker_2`**`(`*`**largs`*`)`

Extrapolation scheme for correlation energies with two adjacent zeta-level bases. Used by `wrappers.complete_basis_set()`.

$$E_{corl}^X = E_{corl}^\infty + \beta X^{-3}$$

CHAPTER

EIGHT

FRACTIONAL OCCUPATION

frac.**frac_nuke**(*mol*, ***kwargs*)

frac.**frac_traverse**(*mol*, ***kwargs*)

frac.**ip_fitting**(*mol*, *omega_l*, *omega_r*, ***kwargs*)

CHAPTER

NINE

BEGINNER PSITHON PROGRAMMING

Note: No recompile of the PSI program is necessary for changes made to files in `$PSIDATADIR`, including those described below.

9.1 Defining a Method Alias

Since quantum chemical methods in PSI4 are accessed through Python functions, and most important quantities are available as PSI variables, it is straightforward to create aliases to commonly run calculations or to define hybrid methods. The `$PSIDATADIR/python/aliases.py` file is intended for editing by the user for this purpose.

As an example, the MP2.5 method is the average of MP2 and MP3. The latter is available through the arbitrary order MPn code and returns all lower energies along with it in PSI variables. The following is basic code that will compute and return the MP2.5 energy.

```
def run_mp2_5(name, **kwargs):

    energy('mp3', **kwargs)
    e_scf = PsiMod.get_variable('SCF TOTAL ENERGY')
    ce_mp2 = PsiMod.get_variable('MP2 CORRELATION ENERGY')
    ce_mp3 = PsiMod.get_variable('MP3 CORRELATION ENERGY')

    ce_mp25 = 0.5 * (ce_mp2 + ce_mp3)
    e_mp25 = e_scf + ce_mp25

    print """  MP2.5 total energy:                      %16.8f\n""" % (e_mp25)
    print """  MP2.5 correlation energy:                %16.8f\n""" % (ce_mp25)

    return e_mp25
```

Compare the above to the method that resides in `aliases.py`. The rationale for the changes is indicated in the comments below.

```
def run_mp2_5(name, **kwargs):
    lowername = name.lower()  # handy variable with name keyword in lowercase
    kwargs = kwargs_lower(kwargs)  # removes case sensitivity in keyword names

    # Run detci calculation and collect conventional quantities
    energy('mp3', **kwargs)
    e_scf = PsiMod.get_variable('SCF TOTAL ENERGY')
    ce_mp2 = PsiMod.get_variable('MP2 CORRELATION ENERGY')
    ce_mp3 = PsiMod.get_variable('MP3 CORRELATION ENERGY')
    e_mp2 = e_scf + ce_mp2  # reform mp2 and mp3 total energies for printing
    e_mp3 = e_scf + ce_mp3

    # Compute quantities particular to MP2.5
    ce_mp25 = 0.5 * (ce_mp2 + ce_mp3)
    e_mp25 = e_scf + ce_mp25
    PsiMod.set_variable('MP2.5 CORRELATION ENERGY', ce_mp25)  # add new method's important results
    PsiMod.set_variable('MP2.5 TOTAL ENERGY', e_mp25)         #     to PSI variable repository
    PsiMod.set_variable('CURRENT CORRELATION ENERGY', ce_mp25)
    PsiMod.set_variable('CURRENT ENERGY', e_mp25)  # geometry optimizer tracks this variable, permit
                                                   #     MP2.5 finite difference optimizations
    # build string of title banner and print results
    banners = ''
    banners += """PsiMod.print_out('\\n')\n"""
    banners += """banner(' MP2.5 ')\n"""
    banners += """PsiMod.print_out('\\n')\n\n"""
    exec banners

    tables  = ''
    tables += """  SCF total energy:                        %16.8f\n""" % (e_scf)
    tables += """  MP2 total energy:                        %16.8f\n""" % (e_mp2)
    tables += """  MP2.5 total energy:                      %16.8f\n""" % (e_mp25)
    tables += """  MP3 total energy:                        %16.8f\n\n""" % (e_mp3)
    tables += """  MP2 correlation energy:                  %16.8f\n""" % (ce_mp2)
    tables += """  MP2.5 correlation energy:                %16.8f\n""" % (ce_mp25)
    tables += """  MP3 correlation energy:                  %16.8f\n""" % (ce_mp3)
    PsiMod.print_out(tables)  # prints nice header and table of all involved quantities to output fi

    return e_mp25
```

One final step is necessary. At the end of the `aliases.py` file, add the following line.

```
procedures['energy']['mp2.5'] = run_mp2_5
```

This permits the newly defined MP2.5 method to be called in the input file with the following command.

```
energy('mp2.5')
```

9.2 Creating a Database

A necessary consideration in constructing a database is the distinction between reagents and reactions. A reagent is a single molecular system (may be a dimer) whose geometry you are possession of and whose electronic energy may be of interest. A reaction is a combination of one or more reagent energies whose value you are interested in and a reference value for which you may or may not be in possession of. A few examples follow. In a database of interaction energies, the reagents are dimers and their component monomers (usually derived from the dimer geometry), and the reactions are the dimer less monomers energies. In a database of barrier heights, the reagents are reactants, products, and transition-state structures, and the reactions are the transition-states less minimum-energy structures. Possibly you may have a collection of structures to simply be acted upon in parallel, in which case the structures are both the reagents and the reactions. The role of the database.py file is to collect arrays and dictionaries that define the geometries of reagents (GEOS), their combination into reactions (RXNM & ACTV), available reference values for reactions (BIND), and brief comments for reagents and reactions (TAGL). The journey from reagent geometries to functional database.py file is largely automated, in a process described below.

- **Prepare geometry files** Assemble xyz files for all intended reagent systems in a directory. Follow the rules below for best results. The filename for each xyz file should be the name of the system. lowercase or MixedCase is preferable (according to Sherrill lab convention). Avoid dashes and dots in the name as python won't allow them. If you're determined to have dashes and dots, they must be replaced by other characters in the process_input line, then translated back in the GEOS section; see NBC10.py for an example.

  - The first line for each xyz file should be the number of atoms in the system.
  - The second line for each xyz file can be blank (interpreted as no comment), anything (interpreted as a comment), or two integers and anything (interpreted as charge, multiplicity, and remainder as comment).
  - The third and subsequent lines have four fields: the element symbol and the three cartesian coordinates in angstroms. The atom lines should not contain any dummy atoms (what's the use in cartesian form). For dimer systems, an algorithm is used to apportion the atoms into two fragments; thus the atoms need not be arranged with all fragmentA atoms before all fragmentB atoms. The algorithm will fail for very closely arranged fragments. For dimers, any charge and multiplicity from the second line will be applied to fragmentA (python); charge and multiplicity may need to be redistributed later in the editing step.

- Run script ixyz2database.pl

  Move into the directory where all your xyz files are located. Run the script in place, probably as `$PSIDATADIR/databases/ixyz2database.pl`. It will ask a number of questions about your intended database and generate a python file named for your database. Uppercase is preferable for database names (according to Sherrill lab convention). Note your choice for the route variable for the next step.

- Edit file database.py
  - All routes. Optionally, rearrange the order of reactions in HRXN as this will define the order for the database.
  - All routes. Optionally, add sources for geometries and any reference data to commented lines above the dbse variable.
  - All routes. Optionally, the comment lines of TAGL may be edited in plain text.
  - All routes. Optionally, fill in reference values (in kcal/mol) into BIND.
  - All routes. Optionally, define alternate sets of reference values in the array BIND_ALTREF in the database.py file to be called in a psi4 input file as benchmark='ALTREF' . See S22.py as an example.
  - All routes. Optionally, fill in the least computationally expensive 2-3 reactions into HRXN_SM and the most expensive into HRXN_LG
  - All routes. Optionally, define subsets of reactions in the array SUBSETARRAY = ['reaction', 'reaction'] in the database.py file to be called in a psi4 input file as subset='SUBSETARRAY'. See NBC10.py as an example.
  - All routes. Necessarily (if charge and multiplicity not read in through line2 = cgmp and nor all neutral singlets), assign correct charge and multiplicity to all reagents.
  - Route 3. Necessarily (if any charge and multiplicity specified for the whole reagent is not intended for fragmentA with neutral singlet fragmentB), apportion charge and multiplicity properly between fragmentA and fragmentB. This is not likely necessary if all subsystems are neutral singlets.
  - Route 2. Necessarily, define the reagents that contribute to each reaction by filling in the empty single-quotes in ACTV. Add or delete lines as necessary if for each reaction more or fewer than three reagents contribute. See NHTBH.py as an example.
  - Route 2. Necessarily, define the mathematical contribution of reagents to reactions by adding a number (most often +1 or -1) for each reagent to the RXNM of each reaction. See NHTBH.py as an example.
  - All routes. Necessarily, copy your new database into `$PSIDATADIR/databases`.

CHAPTER

TEN

FUNCTION INTERCALLS

For many of the PSI4 Python functions described above, it makes scientific sense that they could be called in combination. For instance, one could optimize all the reagents in a database or compute a counterpoise-corrected interaction energy with an extrapolated method. The table below outlines permitted intercalls between functions, showing that db(opt(cbs(energy()))) is allowed, while db(cp(energy())) is not. This table is not yet validated for calls with cp().

Caller	**Callee**				
	cp	**db**	**opt**	**cbs**	**energy**
cp		—	Y	Y	Y
db	—		Y	Y	Y
opt	—	—		Y	Y
cbs	—	—	—		Y
energy	—	—	—	—	

- The command db(opt(cbs(energy()))) is actually expressed as `db( . . . , db_func=opt, opt_func=cbs)`. The perhaps expected final argument of `cbs_func=energy` is not necessary since energy() is always the function called by default. Also, the outermost internal function call (`db_func` above can be called as just `func`. Several examples of intercalls between Python functions can be found in sample input *pywrap_all*.
- All keyword arguments are passed along to each function traversed in the Python driver, so there should be no concern for separating them, grouping them, or designating them for a particular function when undertaking a nested calculation. Where the same keyword is used by multiple functions, prefixes are added, e.g., **db_mode** and **opt_mode**.
- Function intercalls should not be used in sow/reap mode.

CHAPTER

ELEVEN

EMBARRASSING PARALLELISM

Many of the tasks automated by Python wrappers consist of a number of independent psi4 calculations and are thus suited to an embarrassingly parallel mode of operation. In Psithon, these have been dubbed sow/reap procedures and have the following general structure.

- Prepare an input file, simply adding `mode='sow'` to the argument list of an available Python function. Run this quick job to produce input files for lengthier calculations.
- According to the instructions in the output file of the above step, run the generated input files in any order on any variety of computers and architectures. This is the time-intensive portion of the calculation.
- The 'sow' stage also produces a *master* input file (with a `mode='reap'` directive). When all the jobs in the above step are completed, place their output files in the same location as the *master* input, and run this last, quick job to collect the results.
- Sow/reap procedures are governed by the **mode** keyword, choices being `'continuous'`, `'sow'`, and `'reap'`. Only `'sow'` is likely to be used by the user, as `'continuous'` is always the default, and input files with `'reap'` are autogenerated.
- Available at present for Database and finite difference operation of Optimize.

Caution: Some features are not yet implemented. Buy a developer a coffee.

- Local options (e.g., `set scf e_convergence 9`) will not get transmitted to the child jobs.
- Array options (e.g., `set states_per_irrep [2, 1]`) will not get transmitted to the child jobs.
- Function intercalls (e.g., db(opt())) are not tested with sow/reap procedures.

CHAPTER

TWELVE

PSITHON PROGRAMMING BEST PRACTICES

- Thy python functions shall always have final argument **kwargs, that they may take in and pass on keywords meant for other functions. Yea, even the run_mcscf(), and run_ccsd() -type functions that have no use for kwargs. The exceptions are python functions that are only helpers called by a driver function.
- Python functions should read the kwargs dictionary and (possibly) add to it. Functions should not pop or remove keywords from kwargs, even those keywords meaningful only to itself. This will ensure that the complete kwargs is available for pickling and sow/reap procedures. The exception is the molecule argument, which is read by the first function that gets ahold of it. This first function activates the molecule and pops it out of kwargs, effectively setting molecule for all subsequent functions. The code below should suffice.

```
# Make sure the molecule the user provided is the active one
if 'molecule' in kwargs:
    activate(kwargs['molecule'])
    del kwargs['molecule']
molecule = PsiMod.get_active_molecule()
molecule.update_geometry()
```

- Preferrably, the python function signature (for functions intended to be called in input files) is `function(name, **kwargs)`. For functions that have other positional keywords, please bundle them into kwargs at earliest convenience (see Database argument db_name for example).
- After the docstring, the first two lines of your function should be the ones below. The first provides a case insensitive handle to the name argument value. The second converts all the kwargs dictionary keys to lowercase versions of themselves, so that input files can be case insensitive.

```
lowername = name.lower()
kwargs = kwargs_lower(kwargs)
```

- Case sensitivity for kwargs dictionary values still needs to be handled. The first line below shows how to convert argument values to lowercase for matching. When not matching a whole value such that regular expressions are needed, the second line below performs a case insensitive match.

```
if (kwargs['db_mode'].lower() == 'continuous'):
if re.match(r'^sapt', name, flags=re.IGNORECASE):
```

- Match boolean keywords (db_cp in the example below) with expressions like the following, which allow case insensitive yes/true/on/1/no/false/off/0 user input. If your argument's value is a derivative level, similarly, use input.der0th, input.der1st, and input.der2nd.

```
if input.yes.match(str(db_cp)):
elif input.no.match(str(db_cp)):
```

- For keywords that might be used in other functions as well as your own, prepend the argument name with a short representation of your function name. For example, there are keywords cp_func, db_func, and opt_func to request what python function, if not energy(), is called by cp(), database(), and optimize().
- Upon checking in a new python file, edit the file `psi4/doc/userman/source/index.rst` and follow the instructions therein that your file may be autodocumented here.
- Write docstrings! For a major function intended for use in input files, start with the skeleton docstring in `psi4/lib/python/example_docstring` and replace anything that looks like `<this>`. For a behind-the-scenes function or if you don't want the bother of dealing with reStructuredText, just write an ordinary docstring. It will get slurped into the documentation in plain text.
- Your python function should follow PEP8 conventions (without the line-length restriction). I'm aiming for files to pass the line below, unless for good reason. The second line is for database Python files.

```
>>> pep8.py -r --ignore=E501 pythonfile.py
>>> pep8.py -r --ignore=E501,E221,E222,E241,E201,E202 databasefile.py
```

- Your python function should not prevent any test case (`make tests`, NOT `make longtests`) from passing. A test case(s) should be written and checked in for any major python function, so that others do not break your code. If most of your work was on the python (as opposed to c++) side, the test case prefix pywrap_ is suggested.
- Be sure to set any new PSI variables through lines like those below. Especially if the function returns an energy, set the 'current energy' variable. This last is needed to communicate with the optimizer.

```
PsiMod.set_variable('MP2.5 CORRELATION ENERGY', ce_mp25)
PsiMod.set_variable('MP2.5 TOTAL ENERGY', e_mp25)
PsiMod.set_variable('CURRENT ENERGY', e_mp25)
```

- Once your python function is fairly stable on its own, it's potential for interoperability with energy()/opt()/cp()/db()/cbs()/etc. should be evaluated. If it makes physical sense that it should work, you should strive to make that interoperability a reality. Some steps:
  - If any interoperability is possible, define an argument xx_func, where xx is a short name for your function. Add near the top of your function code like the below (less the final two lines). The net result of this code is that if the user specifies no *_func arguments, then energy() gets called. If the user defines xx_func, then its value gets called. If the user defines func, then its value gets reassigned to xx_func, func itself is deleted, and xx_func() gets called. Whatever is getting called is stored in func within the function.

```
# Establish function to call
if not('xx_func' in kwargs):
    if ('func' in kwargs):
        kwargs['xx_func'] = kwargs['func']
        del kwargs['func']
    else:
        kwargs['xx_func'] = energy
func = kwargs['xx_func']
if not func:
    raise ValidationError('Function \'%s\' does not exist to be called by wrapper counterpoi
if (func is db):
    raise ValidationError('Wrapper xx is unhappy to be calling function \'%s\'.' % (func.__r
```

  - If specific interoperabilities are known, code them in. For example, if xx shouldn't call db, add the last two lines above to the xx function. If db shouldn't call xx, add the following two lines below to the db function.

```
if (func is xx):
    raise ValidationError('Wrapper database is unhappy to be calling function \'%s\'.' % (fu
```

- Create a multipart test case that runs some intercalls between your function and others (akin to pywrap_all). In trials, permute the order of calls a few times to expose any calls that don't clean up after themselves and need further attention.
- When all is validated, add your findings to the great interoperability table in the documentation.

CHAPTER

THIRTEEN

EXPERT: PSIMOD MODULE

CHAPTER

FOURTEEN

EXPERT: PYTHON MODULES

14.1 aliases

Module with functions that call upon those in modules `proc`, `driver`, and `wrappers`.

Place in this file quickly defined procedures such as

- aliases for complex methods
- simple modifications to existing methods

`aliases.`**`run_mp2_5`**`(`*name*, ***kwargs*`)`

Function that computes MP2.5 energy from results of a DETCI MP3 calculation.

```
>>> energy('mp2.5')
```

`aliases.`**`run_plugin_ccsd_serial`**`(`*name*, ***kwargs*`)`

Function encoding sequence of PSI module and plugin calls so that Eugene DePrince's ccsd_serial plugin can be called via `driver.energy()`.

```
>>> energy('plugin_ccsd_serial')
```

`aliases.`**`run_plugin_omega`**`(`*name*, ***kwargs*`)`

Function encoding sequence of PSI module and plugin calls, as well as typical options, to access Rob Parrish's omega plugin.

```
>>> energy('plugin_omega')
```

`aliases.`**`sherrillgroup_gold_standard`**`(`*name='mp2'*, ***kwargs*`)`

Function to call the quantum chemical method known as 'Gold Standard' in the Sherrill group. Uses `wrappers.complete_basis_set()` to evaluateo the following expression. Two-point extrapolation of the correlation energy performed according to `wrappers.corl_xtpl_helgaker_2()`.

$$E_{total}^{\text{Au\_std}} = E_{total,\text{ SCF}}^{\text{aug-cc-pVQZ}} + E_{corl,\text{ MP2}}^{\text{aug-cc-pV[TQ]Z}} + \delta_{\text{MP2}}^{\text{CCSD(T)}}\big|_{\text{aug-cc-pVTZ}}$$

```
>>> energy('sherrillgroup_gold_standard')
```

14.2 driver

Module with a *procedures* dictionary specifying available quantum chemical methods and functions driving the main quantum chemical functionality, namely single-point energies, geometry optimizations, response properties, and vibrational frequency calculations.

`driver.`**`gradient`**`(`*name*, ***kwargs*`)`
Function complementary to optimize(). Carries out one gradient pass, deciding analytic or finite difference.

`driver.`**`hessian`**`(`*name*, ***kwargs*`)`
Function to compute force constants. Presently identical to frequency().

`driver.`**`parse_arbitrary_order`**`(`*name*`)`
Function to parse name string into a method family like CI or MRCC and specific level information like 4 for CISDTQ or MRCCSDTQ.

14.3 input

Module import

`input.`**`process_input`**`(`*raw_input*, *print_level=1*`)`

14.4 molutil

Module with utility functions that act on molecule objects.

`molutil.`**`activate`**`(`*mol*`)`
Function to set molecule object *mol* as the current active molecule.

`molutil.`**`dynamic_variable_bind`**`(`*cls*`)`
Function to bind PsiMod.Molecule class.

`molutil.`**`extract_cluster_indexing`**`(`*mol*, *cluster_size=0*`)`
Function to returns a LIST of all subclusters of the molecule *mol* of real size *cluster_size*. If *cluster_size* = 0, returns all possible combinations of cluster size.

`molutil.`**`extract_clusters`**`(`*mol*, *ghost=True*, *cluster_size=0*`)`
Function to return all subclusters of the molecule *mol* of real size *cluster_size* and all other atoms ghosted if *ghost* equals true, all other atoms discarded if *ghost* is false. If *cluster_size* = 0, returns all possible combinations of cluster size.

`molutil.`**`geometry`**`(`*geom*, *name='default'*`)`
Function to create a molecule object of name *name* from the geometry in string *geom*.

`molutil.`**`new_get_attr`**`(`*self*, *name*`)`
Function to redefine get_attr method of molecule class.

`molutil.`**`new_set_attr`**`(`*self*, *name*, *value*`)`
Function to redefine set_attr method of molecule class.

14.5 physconst

14.6 proc

Module with functions that encode the sequence of PSI module calls for each of the *name* values of the energy(), optimize(), response(), and frequency() function.

`proc.`**`run_adc`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for an algebraic diagrammatic construction calculation.

> **Caution:** Get rid of active molecule lines- should be handled in energy.

`proc.`**`run_bccd`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a Brueckner CCD calculation.

`proc.`**`run_bccd_t`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a Brueckner CCD(T) calculation.

`proc.`**`run_cc_gradient`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a CCSD and CCSD(T) gradient calculation.

`proc.`**`run_cc_response`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a CC2 and CCSD calculation.

`proc.`**`run_ccenergy`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a CCSD, CC2, and CC3 calculation.

`proc.`**`run_dcft`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a density cumulant functional theory calculation.

`proc.`**`run_detci`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a configuration interaction calculation, namely FCI, CIn, MPn, and ZAPTn.

`proc.`**`run_dfcc`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a density-fitted coupled-cluster calculation.

`proc.`**`run_dfmp2`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a density-fitted MP2 calculation.

> **Caution:** Get rid of madness-era restart file

`proc.`**`run_eom_cc`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for an EOM-CC calculation, namely EOM-CC2, EOM-CCSD, and EOM-CC3.

`proc.`**`run_eom_cc_gradient`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for an EOM-CCSD gradient calculation.

`proc.`**`run_libfock`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a calculation through libfock, namely RCPHF, RCIS, RTDHF, RTDA, and RTDDFT.

`proc.`**`run_mcscf`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a multiconfigurational self-consistent-field calculation.

`proc.`**`run_mp2`**(*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a MP2 calculation.

proc.**run_mp2_gradient** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a MP2 gradient calculation.

proc.**run_mp2c** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a coupled MP2 calculation.

proc.**run_mp2drpa** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a MP2-DRPA calculation.

proc.**run_mrcc** (*name*, ***kwargs*)
Function that prepares environment and input files for a calculation calling Kallay's MRCC code.

proc.**run_psimrcc** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a PSIMRCC computation using a reference from the MC-SCF module

proc.**run_psimrcc_scf** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a PSIMRCC computation using a reference from the SCF module

proc.**run_sapt** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a SAPT calculation of any level.

proc.**run_sapt_ct** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a charge-transfer SAPT calcuation of any level.

proc.**run_scf** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a self-consistent-field theory (HF & DFT) calculation.

proc.**run_scf_gradient** (*name*, ***kwargs*)
Function encoding sequence of PSI module calls for a SCF gradient calculation.

proc.**scf_helper** (*name*, ***kwargs*)
Function serving as helper to SCF, choosing whether to cast up or just run SCF with a standard guess. This preserves previous SCF options set by other procedures (e.g., SAPT output file types for SCF).

14.7 procutil

Module with utility functions used by several Python functions.

procutil.**format_kwargs_for_input** (*filename*, *lmode=1*, ***kwargs*)
Function to pickle to file *filename* the options dictionary *kwargs*. Mode *lmode* =2 pickles appropriate settings for reap mode. Used to capture Python options information for distributed (sow/reap) input files.

procutil.**format_molecule_for_input** (*mol*)
Function to return a string of the output of `input.process_input()` applied to the XYZ format of molecule *mol*. Used to capture molecule information for distributed (sow/reap) input files.

procutil.**format_options_for_input** ()
Function to return a string of commands to replicate the current state of user-modified options. Used to capture C++ options information for distributed (sow/reap) input files.

Caution: Some features are not yet implemented. Buy a developer a coffee.
- Does not cover local (as opposed to global) options.
- Does not work with array-type options.

procutil.**get_psifile** (*fileno*, *pidspace='21864'*)
Function to return the full path and filename for psi file *fileno* (e.g., psi.32) in current namespace *pidspace*.

procutil.**kwargs_lower**(*kwargs*)
Function to rebuild and return *kwargs* dictionary with all keys made lowercase. Should be called by every function that could be called directly by the user.

14.8 psiexceptions

Module with non-generic exceptions classes.

exception psiexceptions.**PsiException**
Error class for Psi.

exception psiexceptions.**ValidationError**(*msg*)
Error called for problems with the input file. Prints error message *msg* to standard output stream and output file.

14.9 pubchem

Queries the PubChem database using a compound name (i.e. 1,3,5-hexatriene) to obtain a molecule string that can be passed to Molecule.

```
results = getPubChemObj("1,3,5-hexatriene")

Results is an array of results from PubChem matches to your query.
  for entry in results:
     entry["CID"]          => PubChem compound identifer
     entry["IUPAC"]        => IUPAC name for the resulting compound
     entry["PubChemObj"]   => instance of PubChemObj for this compound

     entry["PubChemObj"].getMoleculeString()   => returns a string compatible
                                                  with PSI4's Molecule creation
```

class pubchem.**PubChemObj**(*cid*, *mf*, *iupac*)

getCartesian()
Function to return a string of the atom symbol and XYZ coordinates of the PubChem object.

getMoleculeString()
Function to obtain a molecule string through getCartesian() or fail.

getSDF()
Function to return the SDF (structure-data file) of the PubChem object.

getXYZFile()
Function to obtain preferentially a molecule string through getCartesian() or a query string otherwise.

name()
Function to return the IUPAC name of the PubChem object.

pubchem.**getPubChemResults**(*name*)
Function to query the PubChem database for molecules matching the input string. Builds a PubChem object if found.

14.10 qmmm

Module with classes to integrate MM charges into a QM calculation.

class qmmm.**Diffuse**(*molecule*, *basisname*, *ribasisname*)

fitGeneral()
Function to perform a general fit of diffuse charges to wavefunction density.

fitScf()
Function to run scf and fit a system of diffuse charges to resulting density.

populateExtern(*extern*)

class qmmm.**QMMM**

addChargeAngstrom(*Q*, *x*, *y*, *z*)
Function to add a point charge of magnitude *Q* at position (*x*, *y*, *z*) Angstroms.

addChargeBohr(*Q*, *x*, *y*, *z*)
Function to add a point charge of magnitude *Q* at position (*x*, *y*, *z*) Bohr.

addDiffuse(*diffuse*)
Function to add a diffuse charge field *diffuse*.

populateExtern()
Function to define a charge field external to the molecule through point and diffuse charges.

14.11 text

Module with utility classes and functions related to data tables and text.

class text.**Table**(*rows=()*, *row_label_width=10*, *row_label_precision=4*, *cols=()*, *width=16*, *precision=10*)
Class defining a flexible Table object for storing data.

absolute_to_relative(*Factor=627.5095*)
Function to shift the data of each column of the Table object such that the lowest value is zero. A scaling factor of *Factor* is applied.

copy()
Function to return a copy of the Table object.

format_label()
Function to pad the width of Table object labels.

format_values(*values*)
Function to pad the width of Table object data cells.

save(*file*)
Function to save string of the Table object to *file*.

scale(*Factor=627.5095*)
Function to apply a scaling factor *Factor* to the data of the Table object.

text.**banner**(*text*, *type=1*, *width=35*)
Function to print *text* to output file in a banner of minimum width *width* and minimum three-line height for *type* = 1 or one-line height for *type* = 2.

text.**print_stderr**(*stuff*)
Function to print *stuff* to standard error stream.

`text.`**`print_stdout`**`(`*stuff*`)`
Function to print *stuff* to standard output stream.

14.12 util

Module with utility functions for use in input files.

`util.`**`compare_integers`**`(`*expected*, *computed*, *label*`)`
Function to compare two integers. Prints `util.success()` when value *computed* matches value *expected*. Performs a system exit on failure. Used in input files in the test suite.

`util.`**`compare_matrices`**`(`*expected*, *computed*, *digits*, *label*`)`
Function to compare two matrices. Prints `util.success()` when elements of matrix *computed* match elements of matrix *expected* to number of *digits*. Performs a system exit on failure to match symmetry structure, dimensions, or element values. Used in input files in the test suite.

`util.`**`compare_strings`**`(`*expected*, *computed*, *label*`)`
Function to compare two strings. Prints `util.success()` when string *computed* exactly matches string *expected*. Performs a system exit on failure. Used in input files in the test suite.

`util.`**`compare_values`**`(`*expected*, *computed*, *digits*, *label*`)`
Function to compare two values. Prints `util.success()` when value *computed* matches value *expected* to number of *digits*. Performs a system exit on failure. Used in input files in the test suite.

`util.`**`compare_vectors`**`(`*expected*, *computed*, *digits*, *label*`)`
Function to compare two vectors. Prints `util.success()` when elements of vector *computed* match elements of vector *expected* to number of *digits*. Performs a system exit on failure to match symmetry structure, dimension, or element values. Used in input files in the test suite.

`util.`**`get_memory`**`()`
Function to return the total memory allocation.

`util.`**`get_num_threads`**`()`
Function to return the number of threads to parallelize across.

`util.`**`set_memory`**`(`*bytes*`)`
Function to reset the total memory allocation.

`util.`**`set_num_threads`**`(`*nthread*`)`
Function to reset the number of threads to parallelize across.

`util.`**`success`**`(`*label*`)`
Function to print a '*label*...PASSED' line to screen. Used by `util.compare_values()` family when functions pass.

14.13 wrappers

Module with functions that call the four main `driver` functions: `driver.energy`, `driver.optimize`, `driver.response`, and `driver.frequency`.

`wrappers.`**`call_function_in_1st_argument`**`(`*funcarg*, ***largs*`)`
Function to make primary function call to energy(), opt(), etc. with options dictionary *largs*. Useful when *funcarg* to call is stored in variable.

`wrappers.`**`drop_duplicates`**`(`*seq*`)`
Function that given an array *seq*, returns an array without any duplicate entries. There is no guarantee of which duplicate entry is dropped.

`wrappers.`**`n_body`**`(`*name*, ***kwargs*`)`

`wrappers.`**`reconstitute_bracketed_basis`**`(`*needarray*`)`
Function to reform a bracketed basis set string from a sequential series of basis sets (e.g, form 'cc-pv[q5]z' from array [cc-pvqz, cc-pv5z]). The basis set array is extracted from the *f_basis* field of a *NEED* dictionary in `wrappers.complete_basis_set()`. Result is used to print a nicely formatted basis set string in the results table.

`wrappers.`**`split_menial`**`(`*menial*`)`
Function used by `wrappers.complete_basis_set()` to separate *menial* 'scftot' into [scf, tot] and 'mp2corl' into [mp2, corl].

`wrappers.`**`tblhead`**`(`*tbl_maxrgt*, *tbl_delimit*, *ttype*`)`
Function that prints the header for the changable-width results tables in db(). *tbl_maxrgt* is the number of reagent columns the table must plan for. *tbl_delimit* is a string of dashes of the correct length to set off the table. *ttype* is 1 for tables comparing the computed values to the reference or 2 for simple tabulation and sum of the computed values.

`wrappers.`**`validate_bracketed_basis`**`(`*basisstring*`)`
Function to transform and validate basis sets for cbs(). A basis set with no paired square brackets is passed through with zeta level 0 (e.g., '6-31+G(d,p)' is returned as [6-31+G(d,p)] and [0]). A basis set with square brackets is checked for sensible sequence and Dunning-ness and returned as separate basis sets (e.g., 'cc-pV[Q5]Z' is returned as [cc-pVQZ, cc-pV5Z] and [4, 5]). Note that this function has no communication with the basis set library to check if the basis actually exists. Used by `wrappers.complete_basis_set()`.

`wrappers.`**`validate_scheme_args`**`(`*functionname*, ***largs*`)`
Function called by each extrapolation scheme in `wrappers.complete_basis_set()`. Checks that all the input arguments are present and suitable so that the scheme function can focus on defining the extrapolation.

CHAPTER

FIFTEEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX