EXP-T

a relativistic multireference coupled cluster program

v 1.6.0

User manual

public version

December 20, 2021

Contents

1 Introduction

The EXP-T program package is designed for the high-precision modeling of molecular electronic structure using the relativistic Fock space coupled cluster method (FS-RCC). Features and program components are listed in Sec. 1.1, 1.2 and 1.3.

The authors will be grateful for any comments or suggestions:
exp-t-program@googlegroups.com
https://groups.google.com/d/forum/exp-t-program

1.1 Electronic structure models

The EXP-T package does not include subroutines for solving the (Dirac-) Hartree-Fock equations and subsequent four-index transformation, so molecular integrals have to be imported from third party electronic structure packages. Currently EXP-T is interfaced to the DIRAC program package [1], thus getting access to the wide variety of Hamiltonians and property operators implemented therein.

Models available in EXP-T:

- single-point energy calculations with any point groups and (nearly) all Hamiltonians, implemented in DIRAC (4c-DC, X2Cmmf, 2c-ECP, non-relativistic);
- ground state energy calculations (the $0h0p$ sector): CCSD, CCSD(T), CCSDT-n (n=1,2,3), CCSDT models;
- the FS-CCSD method for excited states is implemented for the $0h1p$, $1h0p$, $1h1p$, $0h2p$, $2h0p$, $0h3p$ Fock space sectors;
- FS-CC models for excited states accounting for triples (CCSDT-n (n=1,2,3), CCSDT) are implemented for the $0h1p$, $1h0p$, $1h1p$, $0h2p$, $0h3p$ Fock space sectors.

  The summary of all electronic structure models implemented in the EXP-T program system is given below:

FS sector	CCSD	CCSD+T(3)	CCSDT-1	CCSDT-2	CCSDT-3	CCSDT
$0h0p$	+	+	+	+	+	+
$0h1p$	+	t	t	t	t	t
$1h0p$	+	t	t	t	t	t
$1h1p$	+	t	t	t	t	t
$0h2p$	+	t	t	t	t	t
$2h0p$	+	–	–	–	–	–
$0h3p$	t	t	t	t	t	t

"+" – implemented, "–" – not implemented, "t" – currently being tested

- "dynamic" energy denominators shifts as a solution of the intruder-state problem [2];
- Padé extrapolation to the zero-shift limit [3];
- finite-field transition moments calculations [4];

- quasidiabatization of SO-coupled states and SO extraction [2].
- «quasi-natural» spinors (NS) and natural transition spinors (NTS);
- one-electron properties via interface to the OneProp program by L. V. Skripnikov [5].

1.2 Features for high-performance calculations

- OpenMP parallelization (for shared-memory systems);
- parallel calculations on NVIDIA GPUs using the CUDA platform [6].

1.3 Program components

`expt.x`	Input processing + all CC calculations.
`heffman.x`	Manipulations with effective Hamiltonian matrices: Padé extrapolations, finite-field transition moments calculations, quasidiabatization of SO-coupled states.
`mknatorb.x`	Transformation of "quasi-natural" spinors to the AO basis.
`expt2pam.x`	Tool for convenient basis set and ECP input.

1.4 Citation

We kindly ask you to acknowledge any use of the EXP-T program system that results in published material using the following citation:

A. V. Oleynichenko, A. Zaitsevskii, E. Eliav. Towards high performance relativistic electronic structure modelling: the EXP-T program package. *Commun. Comput. Inf. Sci.* 1331, 375 (2020). doi: 10.1007/978-3-030-64616-5_33

In addition, credit must be given to the corresponding papers which describe the implementation and the underlying methodological developments:

- Triples in the low Fock space sectors: [7]
- The $0h3p$ Fock space sector: [8]
- Finite-field transition matrix elements (general): [4]
- Finite-field transition matrix elements (the $0h0p \to 1h1p$ excitations): [9]
- Spin-orbit coupling calculations: [2]
- Dynamic denominator shifts (real): [2]
- Dynamic denominator shifts (imaginary, or their real simulation): [10]
- Padé extrapolation of effective Hamiltonian series: [3]

1.5 Credits

EXP-T is based on ideas and design solutions of the suite of FS-CC programs written by Ephraim Eliav and Uzi Kaldor.

2 Installation from source code

2.1 Compiling EXP-T

EXP-T is currently oriented at Unix-like operating systems.

Tools required for compiling the EXP-T package from source code:

- C and Fortran compilers. Currently supported compiler systems are

`gcc/gfortran`	GNU compilers
`icc/ifort`	Intel compilers

- `cmake` version 3.0.2 or higher and `make` utilities:
  https://cmake.org/
  https://www.gnu.org/software/make/

Optional dependencies:

- Implementation of the BLAS/LAPACK linear algebra libraries (OpenBLAS or Intel MKL are recommended);
- NVIDIA drivers and CUDA Toolkit are required to perform parallel calculations on GPU:
  https://www.nvidia.com/Download/index.aspx?lang=en-us
  https://developer.nvidia.com/cuda-toolkit

Download the EXP-T source code as a `*.tar.xz` file and unpack it:

```
tar xvf expt-X.Y.Z.tar.xz
```

Go to the EXP-T home directory, create the `build` directory and make it the current working dir:

```
mkdir build
cd build
```

Compilation *(Intel compilers)*:

```
CC=icc FC=ifort cmake ..
make
```

Compilation *(GNU compilers)*:

```
CC=gcc FC=gfortran cmake ..
make
```

We recommend specifying compilers explicitly (since the default C compiler may be something other than `gcc`).

The CMake utility will try to locate BLAS/LAPACK libraries on your computer; in case of multiple implementations found you will be asked to choose one of them. If there are no pre-installed libraries on your machine, the internal OpenBLAS will be compiled and linked to EXP-T.

When using Intel MKL together with GNU compilers, you will probably need to explicitly specify the path to the MKL home directory by setting the environment variable `MKLROOT`. For example:

```
export MKLROOT=/opt/intel/mkl
```

In case of successful compilation, the executable file `expt.x` will appear in the `build` directory.

We strongly recommend that compilation is followed by testing (see Sect 2.4).

Do not forget to add the directory containing the binaries to the `PATH` environment variable!

2.1.1 Compiling on Mac OS X

The default C compiler for Mac OS X is `clang`; this compiler was not tested, so we strongly recommend to use `gcc` instead:

```
CC=gcc FC=gfortran cmake ..
make
```

We strongly recommend not to use GNU compilers installed with `Homebrew`. Instead, download the GNU compilers suitable for your version of Mac OS X from:
`http://hpc.sourceforge.net/`

In order to use a BLAS/LAPACK implementation other than the one attached to the EXP-T package, you can use the following manual:
`https://pheiter.wordpress.com/2012/09/04/howto-installing-lapack-and-blas-on-mac-os/`
However, the Netlib implementation described there can perform several times slower than the OpenBLAS implementation included in EXP-T.

2.2 Compiling with CUDA

Compilation of EXP-T does not require the user to have a CUDA-compatible GPU device. If the CUDA toolkit is missing, the CUDA code will simply be excluded from the compilation process.

EXP-T is adapted for parallel calculations on NVIDIA GPUs (the CUDA technology [6]).
Notes:

1. At the moment, the OpenMP+CUDA hybrid model as well as multi-GPU support is not implemented. Only one GPU can be used for calculations.
2. non-professional GPUs for gaming are not suitable for CUDA calculations (except for Kepler cards).
3. CUDA code in EXP-T was tested only for CUDA 9.1 and 10.0.
4. Only Intel and GNU compilers were tested for compatibility with CUDA.

To compile EXP-T with CUDA:

1. Make sure your graphics card supports CUDA. CUDA-compatible GPUs are listed here:
   `https://developer.nvidia.com/cuda-gpus`
2. Install proprietary NVIDIA drivers (not the `nouveau` driver!).
   `https://www.nvidia.com/Download/index.aspx?lang=en-us`.
   For Linux you can find drivers in your distribution's repository.
3. Install CUDA toolkit
   `https://developer.nvidia.com/cuda-downloads`
   or from distribution's repository.

4. Make sure that all drivers and CUDA toolkit are installed and work correctly. Compile the test suite in the `samples` directory of the CUDA home directory and run some of them. The most important are the tests employing the CUBLAS linear algebra library.
5. Compile EXP-T as described in Sec. 2.1. All CUDA tools will be located automatically by CMake.
6. Make sure that the results produced by EXP-T do not depend on the use of CUDA.

2.3 Troubleshooting

This section describes the most common problems which can occur during the compilation process. In case of any questions or problems, please don't hesitate to contact us. Don't forget to attach the output file (in case of problems with calculations) or the file with the output of the `make VERBOSE=1` command (in case of problems with compilation).

Google Groups:
`https://groups.google.com/d/forum/exp-t-program`

E-mail:
`exp-t-program@googlegroups.com`

2.3.1 Outdated version of CMake

We highly recommend using CMake version 3.0.2 or higher, otherwise, problems may occur when building the internal OpenBLAS. The following command will show what version of CMake is installed:

```
cmake --version
```

If CMake version is lower than 3.0.2, download the binary distribution from their official website `https://cmake.org/download/`. After installation do not forget to add the directory containing the CMake binary files to the `PATH` environment variable.

Installation of CMake binary files does not require root privileges.

2.4 Testing

Test suite is located in the `test` directory. The testing system is written in the Python programming language (must be pre-installed on your machine). Either Python 2 or Python 3 can be used.

In order to run the testing suite, change the working directory to `test` and run the `test.py` script:

```
cd test
python test.py
```

It is also possible to run an abridged version of the testing suite, that includes only the most important tests. In that case specify the additional argument `quick`:

```
python test.py quick
```

3 Running EXP-T

See Sec. 6 for a description of an additional utitily programs included in the EXP-T package.

A typical calculation consists of two sequential steps: (1) SCF and integral transformation within DIRAC and (2) CC calculation within EXP-T.

3.1 Precomputing spinors and molecular integrals: DIRAC step

The interface to DIRAC was tested only for the DIRAC17 and DIRAC18 releases.

DIRAC stores transformed molecular integrals in the following binary files:

`MRCONEE`	information about SCF calculation, symmetry, occupation numbers, one-electron integrals;
`MDCINT`	symmetry-unique nonzero two-electron integrals;
`MDPROP`	one-electron property integrals.

Run DIRAC and save the files containing the transformed integrals to the working directory using the `--get` option:

```
pam --noarch --mol=<mol-file> --inp=<inp-file> --get="MRCONEE MDCINT MDPROP"
```

If DIRAC starts in parallel mode (MPI), it is necessary to choose the transformation algorithm "scheme 4". By default, DIRAC uses the "scheme 6" algorithm which produces multiple `MDCINT` files [11] and currently EXP-T requires that all integrals are stored in a single file. In order to accomplish this, add the following lines to the DIRAC input file:

```
**MOLTRA
.SCHEME
4
```

To calculate approximate transition moments and intensities without resorting to the finite-field scheme transformed dipole moment integrals are required. We recommend explicitly ask DIRAC to calculate and transform these integrals by adding the `.PRPTRA` and `.DIPOLE` keywords to DIRAC input files:

```
**MOLTRA
.PRPTRA
 . . . . . . .
**PROPERTIES
.DIPOLE
```

If the molecular-mean-field X2C Hamiltonian (X2Cmmf) was used at the DIRAC stage, one ***must*** *specify the* ***`x2cmmf`*** *flag in the EXP-T input file! (see also page 20).*

3.2 Coupled cluster: EXP-T step

Run an EXP-T job using the following command:

```
expt.x [options] <input-file>
```

The EXP-T output (including error messages) is flushed to the UNIX standard output (`stdout`). Use the output redirection operator ">" in case of long time calculations:

```
nohup expt.x <input-file>  >  <output-file> &
```

Examples of DIRAC and EXP-T input files are given in Sec. 5.

3.3 Command-line arguments

Use the `expt.x --help` command to print a list of available command-line arguments:

```
Usage: expt.x [OPTION...] <input-file>
expt -- A Fock-Space Multireference Relativistic Coupled-Cluster Program

  -n, --no-clean             Do not clean scratch directory on exit
                             (use this option to keep cluster amplitudes etc)
  -s, --scratch=PATH         Path to scratch directory (default: ./scratch)
  -?, --help                 Give this help list and exit
      --usage                Give a short usage message and exit
  -V, --version              Print program version and exit

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Please report bugs to <exp-t-program@googlegroups.com>.
```

4 Input files syntax and keywords

See Sec. 6 for the description of additional utitily programs included in the EXP-T package.

The input file format may change in future versions.

The EXP-T input file contains a list of instructions. Each instruction is placed on a separate line and consists of a keyword and a list of arguments. Single-line comments start with `'#'` and continue until the end of the line. Input is case insensitive. Examples of EXP-T input files are given in Sec. 5. Molecular geometry, basis set etc have to be described only once in the DIRAC input files at the molecular integrals calculation stage (not in EXP-T input files).

The detailed description of the keywords used in the EXP-T input file is provided below. For convenience, keywords are grouped into several sections. The order of keywords in the input file can be arbitrary.

General

`title` optional comment.

Syntax: `title <quoted-string>`

Default: no title

Example: `title "my first ccsd calculation"`

`print` print level.

Syntax: `print ( low || medium || high || debug )`

Default: `print medium`

`degen_thresh` energy levels are considered as degenerate if the energy gap between them is lower than `degen_thresh`.

Syntax: `degen_thresh <real thresh>`

Units: atomic

Default: `1e-8`

Model

`sector` specifies the target Fock space sector. To the moment the $0h0p$, $0h1p$, $1h0p$, $1h1p$, $0h2p$, $2h0p$, $0h3p$ FS sectors are implemented (h – holes, p – particles).

Syntax: `sector <integer H>h<integer P>p`

Default: `sector 0h0p` (vacuum)

`model` CC model (approximation to the cluster operator).

Syntax: `model ( ccs || ccd || ccsd || ccsd(t) || ccsd+t(3) || ccsdt-1 || ccsdt-2 || ccsdt-3 || ccsdt )`

Default: `model ccsd`

Notes:

- Keywords `ccsd(t)` and `ccsd+t(3)` are equivalent.
- The codes implementing the CCSD+T(3), CCSDT-n and CCSDT models in the non-trivial sectors and all CC models in the $0h3p$ sector are experimental (currently being tested), to be used with care.
- The current implementation of the CCSD(T) model for the $0h0p$ sector is not very computationally efficient.
- The CCS and CCD models are implemented simply by setting all T_2 (CCS) or T_1 (CCD) amplitudes to zero, this does not make computation faster, and exist primarily for the purposes of testing and debugging.

`occ` spinor occupation numbers. This keyword can be used to setup calculations with a "high-spin" vacuum state [12]. This keyword is alternative to the `occ_irreps` keyword (see below).

Syntax: `occ <list of the 0 and 1 digits separated by spaces>`

Default: occupation numbers are read from MO integrals files

Example: 5 electrons and 10 spinors:
`occ 1 1 0 0 0 1 1 1 0 0`

CC calculations with high-spin vacuum states were tested for the $0h0p$ FS sector only.

`occ_irreps` overall number of electrons in each irrep. Overall irrep occupations are ignored if occupation numbers of individual spinors are given (by the `occ` keyword, see above).

Syntax: `occ_irreps <list of integers>`

Default: occupation numbers are read from MO integrals files

Active space

`nacth/nactp` active (i.e. valence) space specification (by the overall number of active spinors, regardless of their symmetry). This method of configuring the active space is recommended.

- `nacth` – number of highest occupied spinors *(active holes)*
- `nactp` – number of lowest virtual spinors *(active particles)*

It is convenient to setup the active space by the `nacth/nactp` keywords when exploring electronic states of molecules in a range of geometries (potential energy surfaces, transition moment functions etc). Using the `nacth/nactp` keywords one can specify either overall number of active spinors (holes or particles) or number of active spinors in each irreducible representation.

Syntax: `nacth <integer dim>`

`nactp <integer dim>`

or

`nacth <pairs irrep:number-of-spinors>`

`nactp <pairs irrep:number-of-spinors>`

Default: `nacth 0`, `nactp 0` (no active spinors).

Example: consider the low-lying electronic states of the La atom. Those are dominated by the $6s^n5d^m$ configurations, generating even electronic states. So it seems reasonable to include the $6s_{1/2}$, $6d_{3/2}$ and $6d_{5/2}$ atomic spinors of the La^{3+} cation into the active space and achieve target electronic states in the $0h3p$ Fock space sector. In order to make the input file easy to read, one can use two sequential `nactp` directives, one for the "negative" projections of m_j, and one for the "positive" ones:

```
nactp [1/2g+]:3 [3/2g+]:2 [5/2g+]:1
nactp [1/2g-]:3 [3/2g-]:2 [5/2g-]:1
```

One can extend the scope of achievable electronic states by inclusion of the $6p$ and $4f$ spinors:

```
nactp [1/2u+]:4 [3/2u+]:3 [5/2u+]:2 [7/2u+]:1
nactp [1/2u-]:4 [3/2u-]:3 [5/2u-]:2 [7/2u-]:1
```

`active` active (i.e. valence) space specification by energy range. Active spinors must have one-electron energies in the range $\varepsilon_i \in [\varepsilon_{min}; \varepsilon_{max}]$ (regardless of their symmetry). Setting the active space using the `active` keyword is convenient only for atomic calculations.

Syntax: `active energy <real eps_min> <real eps_max>`

Default: no active spinors

Example: `active energy -1.0 5.0`

Properties

`nohermit` do not perform symmetric orthogonalization of model vectors. Orthogonalized model vectors are used only to calculate properties and approximate density matrices; matrices of effective Hamiltonians flushed to HEFF files (see Sec. 6.1) remain non-Hermitian.

Syntax: `nohermit`

Default: "hermitization" is enabled.

`mstdm` enable calculations of the model-space approximations to transition dipole moments (wavefunctions are represented by model vectors) [13, 14, 9]). To avoid huge outputs, use option `nroots` to restrict the set of target roots.

Note that left model vectors are used in bra:

$$d_{if,\eta} = \langle \tilde{\psi}_i^{\perp\perp} | d_\eta | \tilde{\psi}_f \rangle \qquad \eta = x, y, z$$

and if "hermitization" of the effective Hamiltonian matrix is disabled, the TDM matrix will be (normally slightly) non-Hermitian ($|d_{if}| \neq |d_{fi}|$).

This approach typically overestimates transition moments by 30-50%, hence we recommend using it only for semiquantitative estimations, e.g. detection of the most intensive transitions and discerning between 0^+ and 0^- states (such transitions are strictly forbidden).

Syntax: `mstdm`

Default: disabled.

`mdprop` enable calculations of the model-space approximations to arbitrary properties presented in the DIRAC's `MDPROP` unformatted file. To avoid huge outputs, use option `nroots` to restrict the set of target roots. To be used for semiquantitative estimations only.

Syntax: `mdprop "<property-name>" [transpose]`

Default: disabled.

Example: calculation of x,y,z-components of transition moments.

```
mdprop "XDIPLEN"
mdprop "YDIPLEN"
mdprop "ZDIPLEN"
```

`txtprop` enable calculations of the model-space approximations to arbitrary properties. Property matrix is loaded from two formatted files. To avoid huge outputs, use option `nroots` to restrict the set of target roots. To be used for semiquantitative estimations only.

Syntax: `txtprop <string file_re> <string file_im> [transpose]`

Default: disabled.

Example:

```
mdprop PropIntsRe.txt PropIntsIm.txt
```

The code for properties is currently experimental; please, we kindly ask you to report all failures and obviously incorrect results.

`natorb` calculation of "quasi-natural" spinors (NS) and transition spinors (NTS).

Note: density matrices $d_{pq}^{IJ} = \langle \tilde{\psi}_I^{\perp\perp} | a_p^\dagger a_q | \tilde{\psi}_J \rangle$ are constructed with model vectors instead of true wavefunctions.

Syntax: `natorb [<rep>:<state> | <rep1:state1>-<rep2:state2>] ...`

Electronic states are given by the `<rep>:<state>` pair, where:

`<rep>` name of representation in square braces. See Sec. C for the list of representation names used in EXP-T.

`<state>` sequential number of the state in the irrep (1, 2, ...).

Pair of electronic states separated by hyphen shows that NTS between these states are to be calculated. You can ask EXP-T to calculate NS and NTS for arbitrary number of states (or pairs of states).

Default: calculation of NS and NTS is disabled.

Examples:

- NS for the ground and the first excited states of the RaCl molecule:

  `natorb [1/2+]:1 [1/2+]:2`

- NTS for the $(1)0^+$ – $(2)0^+$ transition in the RbCs molecule:

  `natorb [0]:1-[0]:3`

  (note that in the latter example the `[0]:2` state is excluded, since it is the $(1)0^-$ state).

Iterative solution and convergence

`arith` allows one to force EXP-T to use the complex arithmetic everywhere.

Syntax: `arith complex`

`nroots` specifies the number of roots (electronic states) of interest in each irrep (in the target Fock space sector). Model vectors analysis will be performed only for the `nroots` lowest states.

Syntax: `nroots <pairs irrep:number-of-roots>`

Default: all roots are of interest.

Example: consider the electronic states of the RbCs molecule, corresponding to the first three dissociation limits (see, for example, [2, 15] and references therein). In the Hund's case *a* these electronic states are $X^1\Sigma^+$, $a^3\Sigma^+$, $b^3\Pi$, $A^1\Sigma^+$, $c^3\Sigma^+$, $B^1\Pi$. In the Hund's case *c* these states are classified according the projection of the total electronic angular momentum and can be rearranged as follows: (1-3)0^+, (1-3)0^-, (1-4)1, (1)2 states. The names of irreps used in EXP-T are given in Appendix C (the point group of RbCs is $\overline{C_{\infty v}}$). All output and analysis of model vectors will be restricted to these 11 states using the following command:
`nroots [0]:6 [1+]:4 [1-]:4 [2+]:1 [2-]:1`
(note that as in DIRAC 0^+ and 0^- states belong to the same irrep).

`roots_cutoff` specifies the upper energy bound for the electronic states to be analyzed and printed. Model vectors analysis and properties calculations will be performed only for the states with energy lower than `roots_cutoff`. Allowed units of energy are eV, cm^{-1} and Hartrees.

Syntax: `roots_cutoff <real max_energy> ( ev || cm || au )`

Default: all roots are of interest.

Example: `roots_cutoff 30000 cm`

`maxiter` maximum number of iterations allowed to solve the CC equations.

Syntax: `maxiter <integer max>`

Default: `maxiter 50`.

`conv` specifies the requested convergence threshold (in amplitudes):

$$\left| |t_K^{(n+1)}| - |t_K^{(n)}| \right| < conv \quad \forall K \quad \rightarrow \text{converged}$$

Syntax: `conv <real thresh>`

Default: `1e-9`.

`div_thresh` specifies the divergence threshold (in amplitudes). Solution of amplitude equation is diverged if for some t_K holds:

$$\left| |t_K^{(n+1)}| - |t_K^{(n)}| \right| > div\_thresh \quad \rightarrow \text{diverged}$$

May be useful if few initial steps of the iterative solution are divergent.

Syntax: `div_thresh <real thresh>`

Default: `1e+3`.

`diis` enables the DIIS extrapolation technique [16, 17].

Syntax: `diis [ ( off || <integer dim> ) ]`

Disable DIIS: `diis off`

Enable DIIS for triples: `diis triples`

Default: `diis 10` (DIIS is enabled for singles and doubles, disabled for triples)

It is highly recommended to enable DIIS for triples if the CCSDT-n or CCSDT models are employed.

`damping` enable damping in the given FS sector. Amplitudes obtained at the n-th step will be mixed with amplitudes from the $(n-1)$-th step using the damping factor α:

$$T^{(n)'} = \alpha T^{(n-1)} + (1-\alpha)T^{(n)}$$

Damping can aid the convergence of equations in some non-pathological cases (when the lack of convergence is caused by amplitude oscillations rather than intruder states). Note that in other cases damping can significantly slow down convergence.

Syntax: `damping <H>h<P>p <integer last_step> <real factor>`

`<H>h<P>p`	FS sector in which these damping parameters will be applied
`last_step`	at which iteration damping will be disabled
`factor`	damping factor α. Can be negative.

Default: disabled in all sectors

Example: `damping 0h2p 100 0.5`

`shifttype` indicates which formula for denominator shifts should be used [2, 3]. See Sec. **??** for details.

`shifttype` is the same for all FS sectors. If `shifttype` is not equal to `none`, you must specify the shift parameters for at least one FS sector (see keyword `shift`; except for $0h0p$).

Syntax: `shifttype ( none || real || realimag || imag || taylor )`

Possible values:

`none`	no shift
`real`	real shift $$D_K^{'} = D_K + S\left(\frac{S}{D_K+S}\right)^n \quad (1)$$
`realimag`	real simulation of imaginary shift [10] $$D_K^{'} = D_K + \frac{S^2}{D_K}\left(\frac{S^2}{D_K^2+S^2}\right)^n \quad (2)$$
`imag`	imaginary shift [10] $$D_K^{'} = D_K + iS\left(\frac{\lvert S\rvert}{\lvert D_K+iS\rvert}\right)^n \quad (3)$$
`taylor`	extrapolated intermediate Hamiltonian-like shift [18] $$D_K^{'} = (D_K+S)\frac{\left(1-\frac{S}{D_K+S}\right)}{1-\left(\frac{S}{D_K+S}\right)^{n+1}} \quad (4)$$

Default: `shifttype none`

Shift is never applied in the 0h0p sector!

`orbshift` indicates the use of sector-universal shifts. For given "orbital" shift s and cluster amplitude t_K the resulting shift S is defined as

$$S = n_v(t_K) \times s/2,$$

where $n_v(t_K)$ is the number of valence indices of t_K. Thus, it is enough to set only one parameter, s, for all sectors and cluster operators of arbitrary excitation rank. You can also set the attenuation parameter n, see eqs. (1) and (4).

Syntax: `orbshift [<integer power default 3>] <real shift s>`.

Notes:

- The `shift` directive disables orbital shifts.
- The `shifttype` directive is required.

Example:

```
shifttype real
orbshift 3 -0.3   # power == 3, s = -0.3
# shifts for T2 amplitudes will be:
#   1 valence index:   S = s/2 = -0.15
#   2 valence indices: S = s = -0.3
#   3 valence indices: S = 3*s/2 = -0.45
```

`shift` sets the denominator shift parameters in the $n_h h\ n_p p$ FS sector.

`shift <H>h<P>p <int n> <real shift_S1> <real shift_S2> [<real shift_S3>]`

`n`	attenuation parameter (see eqs. (1) and (4));
`shift_S1`	shift value for the $S_1^{n_h h\ n_p p}$ cluster operator (singles). Must be omitted in the $2h0p$ and $0h2p$ FS sectors. It is recommended to set `shift_S1`$=0.0$ in the $1h1p$ sector, unless you really understand the consequences of using nonzero values. Note that `shift_S1` value in this sector does not affect the resulting energies;
`shift_S2`	shift value for $S_2^{n_h h\ n_p p}$ (doubles);
`shift_S3`	shift value for $S_3^{n_h h\ n_p p}$ (triples). Must be used only for CC models including non-perturbative triples.

Default: $n = 3$, all shifts $= 0.0$

Examples:
```
shift 0h1p 3 -0.3 -0.6 -0.9
shift 0h1p 3 -0.6
shift 1h1p 4 0.0 -0.5
```

`reuse` indicates that sorted integrals and cluster amplitudes from the previous calculation should be reused. Cluster amplitudes will be used as an initial guess. If EXP-T fails to locate the files containing amplitudes, they will be recalculated.

Syntax: `reuse <list-of-arguments>`

Possible values of arguments:

`integrals`	load one- and two-electron integrals (from a scratch directory);
`1-integrals`	one-electron integrals only;
`2-integrals`	two-electron integrals only;
`amplitudes`	load amplitudes for all FS sectors occuring in the calculation;
`0h0p, 0h1p, 1h0p, 1h1p, 0h2p, 2h0p, 0h3p`	load amplitudes for the given sector only

Default: no reuse

Note: files with converged amplitudes are flushed to the scratch directory after each calculation (see also Appendix B). To prevent the deletion of files with amplitudes or integrals it is necessary to use the `--no-clean` command-line option.

Example: `reuse integrals 0h0p 0h1p`

`flush` indicates that cluster amplitudes must be flushed to disk every N iterations. Can be useful for very time-consuming calculations with poor convergence. Allows one to change the convergence acceleration strategy.

Syntax: `flush <integer N> iter`

Default: no flush.

Example: `flush 5 iter`

`skip` indicates which sectors will be skipped in the FS-CC calculation. Can be useful in very costly calculations (like CCSDT) in order to avoid recalculation of the first step in already converged sectors. To be used with the `reuse` option.

Syntax: `skip <list-of-sector-symbols>`

Default: no skip.

Example: `skip 0h0p 0h1p`

Interfaces

`integrals` sets paths to files containing transformed molecular integrals.

Syntax: `integrals <string "1-el int-s"> <string "2-el int-s"> [<string "properties int-s">]`

Arguments:

- one-electron integrals file;
- two-electron integrals file;
- properties transformed integrals file.

Default: `integrals MRCONEE MDCINT MDPROP`

Example: `integrals ../MRCONEE-Cinfv ../MDCINT-Cinfv /home/user/MDPROP`

`x2cmmf` since DIRAC's unformatted files do not contain any information about the relativistic Hamiltonian used, the `x2cmmf` flag **must** be used if the molecular-mean-field X2C Hamiltonian (X2Cmmf) was employed at the DIRAC stage.

Syntax: `x2cmmf`

Memory management

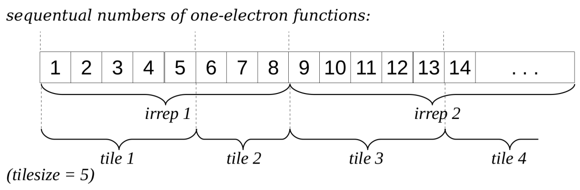
`memory` specifies the amount of memory that EXP-T can use for the job (just for *dynamically* allocated memory).

Syntax: `memory <real size> ( mb || gb )`

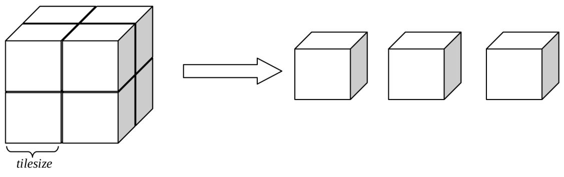
Min value: 10 Mb

Default: `memory 1024 mb` (1 Gb)

`tilesize` the whole set of one-electron functions (spinors or spin-orbitals) is divided into blocks: (1) by irrep; (2) then into sub-blocks with size not exceeding `tilesize` (actually the same as "tiles" in NWChem-TCE):



The N-dimensional arrays of integrals and amplitudes (tensors) also turn out to be divided into blocks containing $\sim (tilesize)^N$ elements:



`tilesize` must be large enough to place at least one N-dimensional array of size $(tilesize)^N$ in RAM. We recommend using as large `tilesize` as possible, otherwise memory management overheads can become enormous. However, it may be useful to decrease the `tilesize` parameter in cases of insufficient RAM or threaded execution (sufficient granularity of arrays leads to better dynamic load balancing among OpenMP threads).

Syntax: `tilesize <integer size>`

Default: `tilesize 100`

`disk_usage` indicates which data should be stored on the disk. Note that very intensive disk usage will slow down calculations; however, disk usage is unavoidable in case of large tasks.

Syntax: `disk_usage <integer mode>`

Possible values of `<mode>`:

`0`	all data is stored in RAM;
`1`	tensors of rank ≥ 6 are stored on disk (T_3 amplitudes etc);
`2`	tensors of rank ≥ 6 and $\langle pp \| pp \rangle$ (4 particles) two-electron integrals are stored on disk;
`3`	tensors of rank ≥ 6, $\langle pp \| pp \rangle$ and $\langle *p \| pp \rangle$ (3 particles) two-electron integrals are stored on disk;
`4`	the same as `3`, + compression of all data. Is suitable for extremely large tasks only.

Default: `2` – $\langle pp||pp \rangle$ and tensors of rank ≥ 6 are stored on disk.

`compress` enables compression of all data written to disk. The LZ4 algorithm [19] is employed (the fastest decompression to the date). Compression can slow down calculations slightly (depends on disk), but the disk space used can be reduced dramatically (up to 3 times).

Syntax: `compress`

Default: disabled (except for `disk_usage=4`)

Parallel execution

`nthreads` number of OpenMP threads. Note that scaling with respect to number of threads is much better for large tasks (400+ spinors).

Syntax: `nthreads <integer n_omp_threads>`

Default: `nthreads 1` (sequential execution).

At the moment parallelization is not used at the integral sorting stage, hence OpenMP can accelerate only the amplitude equations solution stage.

`openmp` the same as `nthreads`.

`cuda` enables parallel calculations on NVIDIA GPU (only a single GPU is supported at the moment). Requires `nthreads 1` (the OpenMP+CUDA hybrid model is not implemented yet).

Syntax: `cuda`

Default: CUDA disabled.

5 Running typical calculations

The input and output files discussed in this section can be found in the `examples` *directory of the EXP-T distro.*

5.1 Ground state energy with CCSD: CO molecule

For input/output files see `examples/CO_ccsd`

Consider the calculation of the CCSD correlation energy of the CO molecule in the cc-pVDZ [20] basis set.

DIRAC input files:

(CO.mol)

```
DIRAC
CO molecule, R = 2.132 bohrs
symmetry C1
C   2    0
        6.    1
C        0.0      0.0      0.0
LARGE BASIS cc-pVDZ
        8.    1
O        0.0      0.0      2.132
LARGE BASIS cc-pVDZ
FINISH
```

(TRA.inp)

```
**DIRAC
.TITLE
CO molecule
.WAVE FUNCTION
.4INDEX
**WAVE FUNCTION
.SCF
*SCF
.CLOSED
14
.ERGCNV
1E-12
**MOLTRA
.ACTIVE
all
*END OF INPUT
```

We run DIRAC with the following command:

```
pam --inp=TRA --mol=CO --get="MRCONEE MDCINT"
```

Note that the `--get` option is used so that `pam` exports the transformed molecular integrals (stored in the `MRCONEE` and `MDCINT` files) from DIRAC's scratch directory to the working directory.

In some cases, it can be convenient to rename the files containing the integrals (this is not mandatory):

```
mv MRCONEE MRCONEE-C1
mv MDCINT MDCINT-C1
```

Now the EXP-T input file `input-C1` is set up:

```
# CO molecule, CCSD/cc-pVDZ
# nonrelativistic, symmetry C1

# task title (of type "string")
title "CO/CCSD/cc-pVDZ"

# print level, default medium
print medium

# max number of iterations
maxiter 30

# convergence threshold (by cluster amplitudes)
conv 1e-9

# target Fock space sector
sector 0h0p

# CC model: ccsd, ccsdt-1, etc
model ccsd

# import integrals from
integrals MRCONEE-C1 MDCINT-C1
```

To run EXP-T, enter:

```
expt.x input-C1
```

The string with the correlation energy can be found at the end of the output. Note that by default EXP-T flushes all output simply to `stdout`.

```
          SCF reference energy =    -112.820480227130
       CCSD correlation energy =      -0.298117912040
             Total CCSD energy =    -113.118598139170
```

5.2 Simple atomic FS-CCSD calculation: the Pb atom

For input/output files see `examples/Pb_fs_ccsd`

We now turn to simple atomic relativistic calculations employing the FS-CCSD approach. Since wave functions of most low-lying electronic states of one-electron atoms and atomic ions are effectively single-reference, it is not very interesting to apply the multireference methodology. In contrast, electronic states of the "two-electron" atoms can hardly be simulated using any other approaches except those based on the multireference wave functions. Consider for example the $6s^2 6p^2$ low-lying states of the Pb atom. The reference state will be Pb^{2+} $6s^2$, and one should select $6p$-spinors as the active ones (overall 6 spinors).

Pb atom is a purely relativistic system, and one should use four-component Hamiltonian or relativistic pseudopotential approach. The latter allows one to incorporate bulk of the Breit interaction and QED corrections. Here the semilocal valence version of the N.S. Mosyagin's RECP for Pb is used (see [21] and references therein).

Input file for the FS-CCSD calculation will be as follows:

```
sector 0h2p
nactp 6
maxiter 500
memory 110 gb
conv 1e-7
disk_usage 0
nthreads 8
model ccsd
```

After the FS-CCSD calculation is finished, one can compare the calculated FS-CCSD excitation energies of Pb with their experimentally measured counterparts (7819, 10650, 21458 and 29467 cm^{-1}, respectively [22]):

```
 Level  Re(eigenvalue)    . . .    Rel eigv, eV  Rel eigv, cm-1  deg  symmetry
 -----------------------  . . .  -----------------------------------------------
@    1    -0.8258538771   . . .    0.0000000000        0.000000   1   0g
@    2    -0.7916687782   . . .    0.9302239316     7502.761992   3   0g 1g+ 1g-
@    3    -0.7784232250   . . .    1.2906537953    10409.824894   5   0g 1g+ 1g- 2g+ 2g-
@    4    -0.7298818976   . . .    2.6115306047    21063.414836   5   0g 1g+ 1g- 2g+ 2g-
@    5    -0.6892150215   . . .    3.7181326781    29988.762480   1   0g
```

The remaining error can be significantly reduced by inclusion of triple excitations [7].

5.3 Electronic states of the HgH^{2+} molecular ion. Energy denominators shifts technique

For input/output files see `examples/HgH2+_shifts`

In almost every problem involving the exploration of potential energy surfaces using the FS-CC method in a fairly wide range of internuclear distances, the intruder state problem arises. The presence of intruder states usually manifests itself as (but does not not simply arise from!)

the appearance of small energy denominators in amplitude equations (see Sec. **??** for details); the presence of small denominators makes the iterative procedure unstable. A straightforward solution of the intruder-state problem can be obtained by modifying (shifting) all energy denominators in the problematic Fock space sector(s) in such a way that the shifts are significant for ill-defined (nearly zero or positive) energy denominators and remain negligibly small for large negative denominators [2, 3]. Note, however, that this does not imply any modification of the right-hand side of the amplitude equations and, thus, introduces an additional (although controllable) approximation.

The denominator shift technique is implemented in EXP-T; different formulas are available (see Sec. 4, keyword `shifttype`). The example discussed below employs the simplest *real* shifts.

Consider the HgH^{2+} molecular ion at $R = 2.0$ bohr. The FS-CC equations cannot be solved at all without using denominator shifts. Let the active space be comprised of the 8 lowest virtual spinors. The EXP-T input file will look like this:

```
# Test:
# (1) HgH2+ ion, FSCC scheme: HgH3+ -> HgH2+
# (2) hamiltionian: 2-comp gatchina ECP
# (3) symmetry Cinfv
# (4) sector 0h1p

title "Test HgH3+ -> HgH2+/gatchina ECP/Cinfv"
maxiter 200
conv 1e-9
active -10 -0.5
sector 0h1p

# import integrals from:
integrals MRCONEE-Cinfv MDCINT-Cinfv

# shifts of denominators:

# real shift:
shifttype real

# shift parameters:
# 1. <n>  -- attenuation parameter (here: 3)
# 3. <S1> -- shift for all S1 amplitudes (here: -0.5)
# 3. <S2> -- shift for all S2 amplitudes (here: -1.0)
shift 0h1p 3 -0.5 -1.0
```

Approaches to the selection of shift parameters are discussed in [2, 3]. You can also perform Pade extrapolation of the series of effective Hamiltonian matrices to the zero shift limit in order to minimize distortions introduced by the shift [3] (see also Sec. 6.1).

5.4 Finite-field transition dipole moments calculations: Rb atom

For input/output files see `examples/Rb_atom_TDMs`

See [14, 23, 4] for detailed discussions on the finite field technique (FF) for transition dipole moments. The algorithm for any FF calculation is as follows:

- solve the HF problem for the non-perturbed Hamiltonian (no external fields);
- transform the molecular integrals to the basis of "non-perturbed" one-electron functions;
- perform two FS-CC calculations with these integrals at external fields strengths $-F$ and $+F$;
- estimate TDMs are estimated using the finite-difference-type formula:

$$\boldsymbol{d}_{\eta,if}^{FF} = (E_i - E_f) \lim_{\substack{F_\eta \to 0 \\ F_{\eta'}=0, \eta' \neq \eta}} \frac{\langle \tilde{\psi}_i^{\perp\perp}(-F_\eta) | \tilde{\psi}_f(+F_\eta) \rangle}{2F_\eta} \quad (\eta = x, y, z), \tag{5}$$

  where $\tilde{\psi}_i$ and $\tilde{\psi}_f$ are model vectors corresponding to **i**nitial and **f**inal electronic states, respectively.

Consider the FF calculation of TDMs for the $^2S_{1/2} \leftrightarrow {}^2P^o_{1/2}$ and $^2S_{1/2} \leftrightarrow {}^2P^o_{3/2}$ transitions in the Rb atom. The basis set and effective core potential employed were taken from [23]; basis set was reduced to $[7s7p5d3f2g]$. The simplest active space required to describe $^2S_{1/2}$, $^2P^o_{1/2}$ and $^2P^o_{3/2}$ states of Rb comprised $5s$- and $5p$-spinors (overall 8 spinors, or 4 Kramers pairs).

Firstly, we perform the Hartree-Fock calculation and subsequent transformation of integrals. Note that transformed integrals of one-electron properties (in this case, the dipole moment operator) are stored in the `MDPROP` unformatted file which should be copied from DIRAC scratch to the working directory:

```
pam --noarch --inp=TRA.inp --mol=Rb.mol --get="MRCONEE MDCINT MDPROP"
```

Other integrals are saved to the `MRCONEE` (one-electron Fock) and `MDCINT` (two-electron) unformatted files. Note that the `.PRPTRA` option as well as the `**PROPERTIES/.DIPOLE` specification is required in the `TRA.inp` input file:

```
**MOLTRA
.PRPTRA
.SCHEME
4
.ACTIVE
energy -100.0 20.0 0.5
**PROPERTIES
.DIPOLE
```

These lines are used to ask DIRAC to calculate and transform dipole moment integrals (these integrals are used later to construct the perturbation due to the external field). Also note that the external uniform electric field lowers spherical symmetry of the atom to $C_{\infty v}$.

Now we are ready to run FSCC calculations in EXP-T. At this step we use the `oneprop` directive to add an external field $\boldsymbol{F}$ contribution to the Fock operator:

$$f'_{pq} = f_{pq} + (d_x)_{pq}F_x + (d_y)_{pq}F_y + (d_z)_{pq}F_z$$

Here we use only the d_z component, stored in the `MDPROP` file in the `ZDIPLEN` record. The operator which is added to the Fock matrix must be fully symmetric. The first EXP-T input file (for field strength $-F$):

```
# title for the task (of type "string")
title "Rb atom -- relativistic EA-CCSD calculation"

# max number of iterations
maxiter 50

# convergence threshold (by cluster amplitudes)
conv 1e-9

# target Fock space sector
sector 0h1p

# active space specification: 8 lowest virtual spinors (5s,5p1/2,5p3/2)
nactp 8

# add the external field contribution to the Fock matrix:
# Fock = Fock + d_z * F_z
# (F_z = 1e-5)
oneprop -1e-5 0.0 "ZDIPLEN"
```

For the other field strength $+F$ it is convenient to reuse two-electron integrals, which are already sorted (the `reuse` option). One-electron integrals depend on the finite perturbation parameter and must be recomputed. Cluster amplitudes converged at $-F$ can also serve as an excellent initial guess for the calculation at $+F$. This will lead to significant computational savings. Use the `--no-clean` option to prevent the deletion of files containing sorted two-electron integrals and converged amplitudes.

The second EXP-T input file (for field strength $+F$):

```
title "Rb atom -- relativistic EA-CCSD calculation"
maxiter 50
conv 1e-9
sector 0h1p
nactp 8

reuse 2-integrals amplitudes
oneprop +1e-5 0.0 "ZDIPLEN"
```

Run EXP-T and save formatted files `HEFF` with effective Hamiltonian matrices to the working directory:

```
expt.x --no-clean input_F- | tee ccsd_F-.out
mv scratch/HEFF HEFF1
expt.x --no-clean input_F+ | tee ccsd_F+.out
mv scratch/HEFF HEFF2
```

The electronic spectrum is given at the end of the EXP-T output file. Corresponding lines begin with the `@` symbol:

```
 Level  Re(eigenvalue)   . . .Rel eigv, cm-1  deg  symmetry
 ----------------------- . . .------------------------------
@    1   -0.1531234146   . . .       0.000000   2   1/2+ 1/2-
@    2   -0.0959179975   . . .  12555.137832   2   1/2+ 1/2-
@    3   -0.0948635671   . . .  12786.558570   2   1/2+ 1/2-
@    4   -0.0948635506   . . .  12786.562182   2   3/2+ 3/2-
```

(compare with experimental values 12578.950 cm^{-1} and 12816.545 cm^{-1} [22]; note that the $^2P^o_{3/2}$ state is split in the external electric field).

Finally, we turn to the calculation of transition moments. The input file for the `heffman.x` utility is as follows (see Sec. 6.1 for details and explanation):

```
file: HEFF1
file: HEFF2
sector: 0h1p
rep: 1
step: 0.00002
print: 1
```

Run `heffman.x`:

```
heffman.x < ff.inp | tee ff.out
```

Transition moments and Einstein coefficients:

```
 transition  energy, cm^-1  . . .    |d|^2       |d|        osc str   A,10^6 s-1
. . .
   1  ->  2     12555.138   . . .  2.627905  1.62108158    0.100220   10.537579
   1  ->  3     12786.559   . . .  5.173064  2.27443710    0.200921   21.911672
   2  ->  3       231.421   . . .  0.000000  0.00001167    0.000000    0.000000
```

Note: this is only the $|d_z|$ component of the full transition moment d. It should be mentioned that the small, but non-zero value of $|d|$ in the last line is an artifact of the FF approach.

Taking into account the degeneracy of the $^2P^o_{3/2}$ state and the spherical symmetry of the Rb atom, the full transition moments $|d|$ and full Einstein coefficients can be calculated:

$$A(^2S_{1/2} \leftarrow {}^2P^o_{1/2}) = 31.6 \text{ s}^{-1}$$

$$A(^2S_{1/2} \leftarrow {}^2P^o_{3/2}) = 32.9 \text{ s}^{-1}$$

(compare with experimental values 34.0 s^{-1} and 37.0 s^{-1} [22]).

Results can be improved significantly by extending the basis set and active space [23].

5.5 «High-spin» CCSD for open shell systems

For input/output files see `examples/O2_highspin`

The example considered in this subsection is borrowed from the DIRAC program manual [12]. Let us calculate the energy of the ground (triplet) state of the O_2 molecule.

The DIRAC input file contains directives for the HF calculation and subsequent integrals transformation:

(O2.mol)

```
test of open-shell CCSD calculation
molecular oxygen at eq distance taken from NIST
automatic symmetry detection: DIRAC will identify the Dinfh group
C   1              A
       8.    2
O      0.0       0.0      0.60376
O      0.0       0.0     -0.60376
LARGE BASIS cc-pVDZ
FINISH
```

(TRA.inp)

```
!
! calculation of the triplet ground state of molecular oxygen
! (Dirac-Coulomb Hamiltonian)
**DIRAC
.TITLE
molecular oxygen -- ground state
.WAVE F
.4INDEX
**WAVE FUNCTIONS
.SCF
*SCF
.CLOSED SHELL
 6 8
.OPEN SHELL
1
2/4,0
*END OF
```

DIRAC writes spinor occupation numbers to the `MRCONEE` unformatted file, but only spinors belonging to closed shell (the `.CLOSED_SHELL` keyword) are labeled as occupied. This is why EXP-T will calculate the energy for the O_2^{2+} cation without manually specified occupation numbers.

EXP-T provides two ways to set the occupation numbers of spinors:

- by setting the number of occupied spinors in each irrep (the `occ_irreps` keyword). For the example under consideration:

```
occ_irreps 3 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 2 2 1 1
```

Note that the $1s$-orbitals of oxygen atoms are frozen here, and the corresponding electrons should not be accounted for in the `occ_irreps` string. In order to set the number of electrons in each irrep correctly, we recommend starting with a calculation for default occupation numbers and finding strings with an irrep order in the output:

```
0 1/2g+
1 1/2g-
2 3/2g+
3 3/2g-
4 5/2g+
5 5/2g-
   ...
16 1/2u+
17 1/2u-
18 3/2u+
19 3/2u-
20 5/2u+
21 5/2u-
    ...
```

For the example discussed:

– 3 electrons occupy the lowest spinors belonging to the `1/2g+` irrep;

– `1/2g-`, `1/2u+`, `1/2u-`: 2 electrons in each irrep;

– `3/2g-`, `3/2u+`, `3/2u-`: 1 electron in each irrep.

- you can also specify the occupation number for *each* spinor explicitly:

```
occ 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 ...
```

To identify spinors which should be populated in the electronic state under consideration, we again recommend starting the calculation with the default occupation numbers, then interrupting the task, finding information about spinors and restarting the calculation with the correct occupation numbers.

Note that, in fact, non-canonical orbitals are used, therefore the Fock matrix is not diagonal and the vacuum determinant energy $\langle \Phi_0 | H | \Phi_0 \rangle$ is not equal to the SCF energy value obtained before:

```
 Total SCF energy =                               -149.686661451842
 ...
 sorting one-electron integrals ...
   Fock matrix reconstruction ...
   SCF energy (energy of reference determinant) =    -149.718144633810 a.u.
   SCF energy (energy of reference determinant) was updated:
     old energy =    -149.686661451842 a.u.
     new energy =    -149.718144633810 a.u.
 ...
          SCF reference energy =    -149.718144633810
       CCSD correlation energy =      -0.366958682652
             Total CCSD energy =    -150.085103316462
```

6 Utility programs

6.1 heffman – manipulations with effective Hamiltonian matrices

The `heffman.x` program is designed to perform manipulations with FS-RCC effective Hamiltonians matrices. Implemented features are:

- the finite-field technique for transition dipole moments calculations [14, 23, 4];
- extraction of spin-orbit interactions from full-relativistic models by projection [2];
- Pade extrapolation of series of effective Hamiltonians.

`heffman.x` reads matrices of effective Hamiltonians from formatted files supported by both EXP-T and the latest versions of DIRAC.

- to compute finite-field transition moments [23, 4] two files with effective Hamiltonians must be provided, calculated at field 1 and field 2;
- to separate spin-orbit interactions by projection, two files with effective Hamiltonians must be provided, the first for spin-orbit (nearly) switched off and the second for spin-orbit switched on.

  In this case you cannot freeze spinors at the FSCC step!
- to perform Pade extrapolation of the series of effective Hamiltonians, you have to put a sequence of effective Hamiltonians to a single file using the `cat` command:

  ```
  cat heff_n_eq_1 heff_n_eq_2 heff_n_eq_3 > some_file_name
  ```

  Recall that each Hamiltonian is calculated at its own attenuation parameter n, see Sec. **??**. You can also perform extrapolation before TDM or SO calculation, just use such "composite" files with effective Hamiltonians.

Additional input file with options must be redirected to `stdin`:

```
heffman.x < input_file
```

`heffman.x` input files are very similar to EXP-T input files, but keywords are separated from their arguments with the ':' symbol, and the symbol '#' at the beginning of a comment must be placed in the first position of the line.

List of keywords:

`file:` file with the first effective Hamiltonian or with a sequence of heff's to be extrapolated, e.g.

- heff for the first field value;
- heff for the spin-orbit interaction (nearly) switched off;
- simply heff you want to get the eigenstates;
- sequence of heff's you want to Pade-extrapolate.

Syntax: `file: <string path-to-first-heff>`

If you are doing finite-field transition moment calculations or SO separation, you have to supply the second file (see the next item).

`file:` heff for another field value, or heff for the spin-orbit switched on, respectively. If there is no second file, the effective Hamiltonian eigenvectors and eigenvalues are written to the formatted file HEFFEVF.

Syntax: `file: <string path-to-second-heff>`

`sector:` the Fock space sector you are interested in (heff files for higher sectors contain several heff matrices).

Syntax: `sector: <H>h<P>p`

Example: `0h1p`, `1h0p`, `2h0p`, `0h2p`, `1h1p`.

`rep:` irreducible representation you are interested in. As a rule, `rep: 1` for the fully symmetrical rep.

Syntax: `rep: <integer n>`

Enumeration of irreps begins from 1!

`main:` for the FF-TDM calculations or Pade extrapolation: simply the number of lowest states or the range of states of given sector and symmetry you want to see in the output file. For the SO extraction by the projected approximant construction [3] this keyword defines the subspace of states used for projection.

Syntax: `main: <integer number>` or `main: <integer number1>-<integer number2>`.

Default: all states.

Example: `main: 10` (equivalently, `main: 1-10`)

`[L/M]` *(optional)* to perform Pade extrapolation, you have to indicate which Pade approximant [L/M] should be constructed ([0/1], [1/1], [0/2] etc). You should have L+M+2 effective Hamiltonian matrices to build the [L/M] approximant. You can cut off all states above subspace done by the `main` keyword (see above); to do this, add the `cut` keyword.

Syntax: `[L/M] [cut]`

Default: Padé extrapolation disabled

Example: `[0/1] cut`

`step:` *only for FF calculations.* Difference of field strenghts (a.u.) in two finite-field calculations. The presence of this line activates the FF TDM calculation mode (do not use this keyword in other types of calculations).

Syntax: `step: <real value>`

`scale:` *only for SO calculations.* Fraction of effective spin-orbit retained in “scalar” calculations to get good relativistic symmetries (typically of order 10^{-4}. Omit this line if you do not want to diabatize anything.

Syntax: `scale: <real value>`

`ground:` ground state energy (a.u.). Useful only when you have to compute energies with respect to some state (normally ground) which belongs to another rep or sector.

Syntax: `ground: <real energy>`

Default: `ground: 0.0`

`print:` print level. `print: 0` or `print: 1` are normally ok; enlarging this number, you can get any amount of superfluous information listed.

Syntax: `print: <integer level>`

Default: `print: 1`

`Bloch` activates the computation of finite-field transition dipoles with Bloch effective Hamiltonian eigenvectors (note: TDM matrices will be non-Hermitian, this has no physical meaning). Otherwise, ff computations are performed with des Cloizeaux effective Hamiltonian eigenvectors ("hermitization").

Syntax: `Bloch`

Default: "hermitization" is enabled in finite-field calculations

When using any of the techniques implemented in the `heffman.x` program, please, cite the corresponding paper:

- finite-field transition dipole moment calculations: [4]
- quasidiabatization by projection and SO extraction: [2]
- Pade extrapolation: [3]

6.2 mknatorb – transformation of quasi-natural spinors to AO basis

Having model vectors $|\tilde{\psi}_i\rangle$ we can construct approximate density $d_{pq} = \langle\tilde{\psi}_i|a_p^\dagger a_q|\tilde{\psi}_i\rangle$ and transition density matrices $d_{pq} = \langle\tilde{\psi}_i|a_p^\dagger a_q|\tilde{\psi}_j\rangle$ and then calculate "quasi-natural" (transition) molecular spinors (NS and NTS, respectively). These operations can be performed by the EXP-T itself (see Sec. 4, keyword `natorb`). EXP-T stores NS and NTS expanded *in the basis of molecular spinors* in formatted files named `NATORB*.dat`. To use visualization tools available in DIRAC one have to transform NS or NTS to the *AO basis* and store them in the DFPCMO-type formatted file. This transformation can be performed by the `mknatorb.x` utility.

To the moment `mknatorb.x` is able to work with real point groups only!

`mknatorb.x` requires three command-line arguments:

- path to the `DFPCMO` formatted file produced by DIRAC (contains molecular spinors expanded in the AO basis);
- path to the `NATORB*.dat` formatted file produced by EXP-T and stored in its scratch directory (contains natural spinors expanded in the basis of molecular spinors);
- name of the `DFPCMO`-like formatted file to be generated (will contain natural spinors expanded in the AO basis).

To ask DIRAC to save molecular spinors (vectors) to the `DFPCMO` file, add the `.PCMOUT` keyword to the `**GENERAL` section of the DIRAC `.inp` input file:

```
**GENERAL
.PCMOUT
```

and then copy this file to the working directory with the `-get` option:

```
pam --inp=<inp-file> --mol=<mol-file> --get="DFPCMO ..."
```

Running `mknatorb.x` (for example):

```
mknatorb.x DFPCMO NATORB_0h1p_1:1.dat FSCCNO
```

Resulting formatted file contains fractional NS occupation numbers instead of one-electron energies. This file is perceived by DIRAC as a file with (Dirac-)Hartree-Fock problem solution; now natural spinors stored in it can be analyzed using DIRAC tools (e.g. densities can be built and visualized). For more information on visualization in DIRAC, see [24].

Appendix A Alphabetical index of keywords

Appendix B Temporary files

During the calculation, EXP-T operates with temporary files stored in the *scratch* directory. We recommend to remove these temporary files after every calculation (except for those planned to be used in the future, like the `HEFF` files or files with cluster amplitudes). All types of temporary files generated by EXP-T are listed and described below.

`HINT` one-electron integrals – core Hamiltonian matrix + one-particle part of perturbation (if presented). Recommended for removal after calculation.

`VINT-*-*-*-*` two-electron integrals. These files are generated during the integral sorting stage. Recommended for removal after calculation.

Approximate number of files: $\sim N_b^4$ (N_b – number of blocks into which the whole set of spinors is divided by symmetry and the ***`tilesize`*** *parameter)*

`*.sb` single symmetry block of molecular integrals/cluster amplitudes. It is just a part of a diagram. During the calculation each file is mapped to some structure in RAM. Recommended for removal after calculation.

Approximate number of files: $\sim 10 \times N_b^4$ for CCSD, $\sim N_b^6$ for CCSDT-n and CCSDT models – ***can be very large!***

`*.dg` file containing diagram. Diagram files containing cluster amplitudes (`t1c.dg`, `t2c.dg` etc) can be used in subsequent calculations when constructing the initial approximation to amplitudes (see Sec. 4, keyword `reuse`).

`HEFF` formatted file containing effective Hamiltonian matrices. These files can be read by the `heffman.x` utility program used to perform Padé extrapolation of series of effective Hamiltonians [3], finite-field transition moments calculations [23, 4], spin-orbit coupling calculations [2].

`*DIPLEN` formatted files with dipole moment integrals (just its electronic part) in the basis of molecular spinors.

`MVCOEF**` binary files containing model vectors expanded in the basis of model determinants. These files are used to organize data flow inside EXP-T in the most convenient and logical manner; model vectors are used for calculation of model-space approximations of TDMs and "quasi-natural" spinors, etc.

`NATORB*.dat` formatted text files containing "quasi-natural" spinors expanded in the basis of molecular spinors. These expansions can be transformed to the AO basis with the `mknatorb.x` utility program (see Sec. 6.2).

Appendix C Symmetry and irreducible representations

This section lists the names of irreducible representations (irreps) used in EXP-T.

- For nonrelativistic groups, irrep is determined by the spatial symmetry (Mulliken notation is used) and the M_s value (for $M_s >= 2$ only symbol 2 is used);
- For relativistic double groups we use Mulliken-type notation (completely inherited from DIRAC);
- For $U(1) = \overline{C_{\infty v}}$ and $U(1) \times C_i = \overline{D_{\infty h}}$ irreps are determined by Ω values.

C_1

```
A_a
A_b
A_-3/2
A_+3/2
A_0
A_4
A_+1
A_-1
```

C_2

```
A_a    B_a
A_b    B_b
A_-3/2 B_-3/2
A_+3/2 B_+3/2
A_0    B_0
A_2    B_2
A_+1   B_+1
A_-1   B_-1
```

C_s

```
A'_a    A"_a
A'_b    A"_b
A'_-3/2 A"_-3/2
A'_+3/2 A"_+3/2
A'_0    A"_0"
A'_2    A"_2"
A'_+1   A"_+1
A'_-1   A"_-1
```

C_i

```
Ag_a    Au_a
Ag_b    Au_b
Ag_-3/2 Au_-3/2
Ag_+3/2 Au_+3/2
Ag_0    Au_0
Ag_2    Au_2
Ag_+1   Au_+1
Ag_-1   Au_-1
```

C_{2v}

```
A1_a    B2_a    B1_a    A2_a
A1_b    B2_b    B1_b    A2_b
A1_-3/2 B2_-3/2 B1_-3/2 A2_-3/2
A1_+3/2 B2_+3/2 B1_+3/2 A2_+3/2
A1_0    B2_0    B1_0    A2_0
A1_2    B2_2    B1_2    A2_2
A1_+1   B2_+1   B1_+1   A2_+1
A1_-1   B2_-1   B1_-1   A2_-1
```

C_{2h}

```
Ag_a    Bg_a    Bu_a    Au_a
Ag_b    Bg_b    Bu_b    Au_b
Ag_-3/2 Bg_-3/2 Bu_-3/2 Au_-3/2
Ag_+3/2 Bg_+3/2 Bu_+3/2 Au_+3/2
Ag_0    Bg_0    Bu_0    Au_0
Ag_2    Bg_2    Bu_2    Au_2
Ag_+1   Bg_+1   Bu_+1   Au_+1
Ag_-1   Bg_-1   Bu_-1   Au_-1
```

D_2

```
A_a    B3_a    B1_a    B2_a
A_b    B3_b    B1_b    B2_b
A_-3/2 B3_-3/2 B1_-3/2 B2_-3/2
A_+3/2 B3_+3/2 B1_+3/2 B2_+3/2
A_0    B3_0    B1_0    B2_0
A_2    B3_2    B1_2    B2_2
A_+1   B3_+1   B1_+1   B2_+1
A_-1   B3_-1   B1_-1   B2_-1
```

D_{2h}

```
Ag_a    B1u_a    B2u_a    B3g_a    B3u_a    B2g_a    B1g_a    Au_a
Ag_b    B1u_b    B2u_b    B3g_b    B3u_b    B2g_b    B1g_b    Au_b
Ag_-3/2 B1u_-3/2 B2u_-3/2 B3g_-3/2 B3u_-3/2 B2g_-3/2 B1g_-3/2 Au_-3/2
Ag_+3/2 B1u_+3/2 B2u_+3/2 B3g_+3/2 B3u_+3/2 B2g_+3/2 B1g_+3/2 Au_+3/2
Ag_0    B1u_0    B2u_0    B3g_0    B3u_0    B2g_0    B1g_0    Au_0
Ag_2    B1u_2    B2u_2    B3g_2    B3u_2    B2g_2    B1g_2    Au_2
Ag_+1   B1u_+1   B2u_+1   B3g_+1   B3u_+1   B2g_+1   B1g_+1   Au_+1
Ag_-1   B1u_-1   B2u_-1   B3g_-1   B3u_-1   B2g_-1   B1g_-1   Au_-1
```

$\overline{C_1}$

```
A
a
```

$\overline{C_2}$

```
1E 2E
a  b
```

$\overline{C_s}$

```
1E 2E
a  b
```

$\overline{C_i}$

```
AG AU
ag au
```

$\overline{C_{2v}}$

```
1E 2E
a  b
```

$\overline{D_2}$

```
1E 2E
a  b
```

$\overline{D_{2h}}$

```
1Eg 2Eg 1Eu 2Eu
ag  bg  au  bu
```

$\overline{C_{\infty v}}$

```
 1/2+  1/2-  3/2+  3/2-  5/2+  5/2-  7/2+  7/2-
 9/2+  9/2- 11/2+ 11/2- 13/2+ 13/2- 15/2+ 15/2-
17/2+ 17/2- 19/2+ 19/2- 21/2+ 21/2- 23/2+ 23/2-
25/2+ 25/2- 27/2+ 27/2- 29/2+ 29/2- 31/2+ 31/2-
    0    1+    1-    2+    2-    3+    3-    4+
   4-    5+    5-    6+    6-    7+    7-    8+
   8-    9+    9-   10+   10-   11+   11-   12+
  12-   13+   13-   14+   14-   15+   15-   16+
```

$\overline{D_{\infty h}}$

```
1/2g+  1/2g-  3/2g+  3/2g-  5/2g+  5/2g-  7/2g+  7/2g-
9/2g+  9/2g- 11/2g+ 11/2g- 13/2g+ 13/2g- 15/2g+ 15/2g-
1/2u+  1/2u-  3/2u+  3/2u-  5/2u+  5/2u-  7/2u+  7/2u-
9/2u+  9/2u- 11/2u+ 11/2u- 13/2u+ 13/2u- 15/2u+ 15/2u-
  0g     1g+    1g-    2g+    2g-    3g+    3g-    4g+
  4g-    5g+    5g-    6g+    6g-    7g+    7g-    8g+
  0u     1u+    1u-    2u+    2u-    3u+    3u-    4u+
  4u-    5u+    5u-    6u+    6u-    7u+    7u-    8u+
```

References

[1] DIRAC, a relativistic ab initio electronic structure program, Release DIRAC18 (2018), written by T. Saue, L. Visscher, H. J. Aa. Jensen, and R. Bast, with contributions from V. Bakken, K. G. Dyall, S. Dubillard, U. Ekstroem, E. Eliav, T. Enevoldsen, E. Fasshauer, T. Fleig, O. Fossgaard, A. S. P. Gomes, E. D. Hedegaard, T. Helgaker, J. Henriksson, M. Ilias, Ch. R. Jacob, S. Knecht, S. Komorovsky, O. Kullie, J. K. Laerdahl, C. V. Larsen, Y. S. Lee, H. S. Nataraj, M. K. Nayak, P. Norman, G. Olejniczak, J. Olsen, J. M. H. Olsen, Y. C. Park, J. K. Pedersen, M. Pernpointner, R. Di Remigio, K. Ruud, P. Salek, B. Schimmelpfennig, A. Shee, J. Sikkema, A. J. Thorvaldsen, J. Thyssen, J. van Stralen, S. Villaume, O. Visser, T. Winther, and S. Yamamoto (see http://diracprogram.org).

[2] A. Zaitsevskii, N. S. Mosyagin, A. V. Stolyarov, and E. Eliav. Approximate relativistic coupled-cluster calculations on heavy alkali-metal diatomics: Application to the spin-orbit-coupled $A^1\Sigma^+$ and $b^3\Pi$ states of RbCs and Cs_2. *Phys. Rev. A*, 96:022516, 2017.

[3] A. Zaitsevskii and E. Eliav. Padé extrapolated effective Hamiltonians in the Fock space relativistic coupled cluster method. *Int. J. Quantum Chem.*, 118(23):e25772, 2018.

[4] A. V. Zaitsevskii, L. V. Skripnikov, A. V. Kudrin, A. V. Oleinichenko, E. Eliav, and A. V. Stolyarov. Electronic transition dipole moments in relativistic coupled-cluster theory: the finite-field method. *Optics and Spectroscopy (English translation of Optika i Spektroskopiya)*, 124(4):451, 2018.

[5] http://qchem.pnpi.spb.ru/Leonid.

[6] https://developer.nvidia.com/cuda-zone.

[7] A. V. Oleynichenko, A. Zaitsevskii, L. V. Skripnikov, and E. Eliav. Relativistic fock space coupled cluster method for many-electron systems: non-perturbative account for connected triple excitations. *Symmetry*, 12(7), 2020.

[8] L. V. Skripnikov, A. V. Oleynichenko, A. V. Zaitsevskii, D. E. Maison, and A. E. Barzakh. Relativistic fock space coupled cluster study of bismuth electronic structure to extract the Bi nuclear quadrupole moment, 2021.

[9] A. Zaitsevskii, A. V. Oleynichenko, and E. Eliav. Finite-field calculations of transition properties by the fock space relativistic coupled cluster method: transitions between different fock space sectors. *Symmetry*, 12(11), 2020.

[10] A. V. Oleynichenko, L. V. Skripnikov, A. Zaitsevskii, E. Eliav, and V. M. Shabaev. Diagonal and off-diagonal hyperfine structure matrix elements in KCs within the relativistic Fock space coupled cluster theory. *Chem. Phys. Lett.*, 756:137825, 2020.

[11] http://diracprogram.org/doc/master/manual/moltra.html#scheme.

[12] http://www.diracprogram.org/doc/release-18/tutorials/highspin_cc/O2.html.

[13] A. Hehn and L. Visscher. *(to be published)*.

[14] A. Zaitsevskii and A. P. Pychtchev. On the finite-field transition dipole moment calculations by effective Hamiltonian methods. *Eur. Phys. J. D*, 4(3):303, 1998.

[15] V. Krumins, A. Kruzins, M. Tamanis, R. Ferber, A. Pashov, A.V. Oleynichenko, A. Zaitsevskii, E.A. Pazyuk, and A.V. Stolyarov. The branching ratio of intercombination $A^1\Sigma^+ \sim b^3\Pi \to a^3\Sigma^+/X^1\Sigma^+$ transitions in the RbCs molecule: measurements and calculations. *J. Quant. Spectrosc. Radiat. Transf.*, 256:107291, 2020.

[16] P. Pulay. Convergence acceleration of iterative sequences. The case of SCF iteration. *Chem. Phys. Lett.*, 73(2):393 – 398, 1980.

[17] G. E. Scuseria, T. J. Lee, and H. F. Schaefer III. Accelerating the convergence of the coupled-cluster approach: the use of the DIIS method. *Chem. Phys. Lett.*, 130(3):236 – 239, 1986.

[18] E. Eliav, M. J. Vilkas, Y. Ishikawa, and U. Kaldor. Extrapolated intermediate Hamiltonian coupled-cluster approach: theory and pilot application to electron affinities of alkali atoms. *J. Chem. Phys.*, 122:224113, 2005.

[19] LZ4 - Fast LZ compression algorithm. http://www.lz4.org.

[20] T. H. Dunning. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. *J. Chem. Phys.*, 90(2):1007, 1989.

[21] Nikolai S. Mosyagin, Andrei V. Zaitsevskii, and Anatoly V. Titov. Generalized relativistic effective core potentials for superheavy elements. *Int. J. Quantum Chem.*, 120(2):e26076, 2020.

[22] J. E. Sansonetti and W. C. Martin. Handbook of Basic Atomic Spectroscopic Data. *J. Phys. Chem. Ref. Data*, 34(4):1559, 2005.

[23] A. A. Medvedev, A. V. Stolyarov, A. Zaitsevskii, and A. Eliav. Relativistic calculations on the electric dipole transition probabilities of the RbAr exciplex. *Nonlinear Phenomena in Complex Systems*, 20(2):205, 2017.

[24] http://www.diracprogram.org/doc/release-18/manual/visual.html.