The PSI4 User's Manual

C. David Sherrill,[a] T. Daniel Crawford,[b] Edward F. Valeev,[b]
Micah L. Abrams,[b] Rollin A. King,[c], Ashley Ringer[a]
and Justin M. Turney[d]

[a] *Center for Computational Molecular Science and Technology, Georgia Institute of Technology, Atlanta, Georgia 30332-0400*

[b] *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061-0001*

[c] *Department of Chemistry, Bethel College, St. Paul, Minnesota 55112-6999*

[d] *Center for Computational Quantum Chemistry, University of Georgia, Athens, Georgia 30605-2556*

PSI4 Version: 4.0.0-alpha
Created on: July 20, 2011
Report bugs to: psicode@users.sourceforge.net

Contents

1 Introduction

1.1 Overview

This manual explains how to use the PSI4 suite of *ab initio* quantum chemical programs. In this section, we provide an overview of some of the features of PSI4 along with the prerequisite steps for running calculations. Section 3 provides a brief tutorial to help new users get started. Section 4 offers further details into the structure of PSI4 input files and a discussion of some of the most important options. 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 will eventually include a description of the input keywords and command-line options for each module, as well as numerous examples of PSI4 input and basis set files. For the latest PSI4 documentation, check www.psicode.org.

The PSI4 package was developed to perform high-accuracy quantum mechanical computations on challenging chemical species and to provide an infrastructure for the development of new theoretical techniques. Hence, it has a very flexible input scheme which allows non-standard computations, and it is easily adapted to enable new capabilities.

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

> T. Daniel Crawford, C. David Sherrill, Edward F. Valeev, Justin T. Fermann, Rollin A. King, Matthew L. Leininger, Shawn T. Brown, Curtis L. Janssen, Edward T. Seidl, Joseph P. Kenny, and Wesley D. Allen, *J. Comput. Chem.* **28**, 1610-1616 (2007).

1.2 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 the PSI4 Installation Manual, available as part of the package or at the PSI4 website above.

1.3 Supported Architectures

This section needs to be updated.

The majority of PSI4 was developed on x86/GNU Linux workstations. The complete list of tested architectures to which PSI4 has been ported is shown in Table 1. If you don't find your system in the Table, there's a good chance that you will be able to install PSI4 on your system if you have the prerequisite tools and math and utility libraries described in the installation manual.

Table 1: Platforms on which PSI4 has been installed successfully.

Architecture	Notes
Compaq Alpha Tru64 UNIX	64-bit mode
IBM AIX 4.3.3, 5.x on PowerPC	64-bit mode
Linux on Intel/AMD x86 and x86-64	32 and 64-bit
Apple OS X (Darwin) on PowerPC and Intel	
SGI IRIX64 (>6.5.15)	64-bit

Table 2: Summary of theoretical methods available in PSI4.

Method	Energy	Gradient	Hessian
RHF SCF	Y	Y	Y
ROHF SCF	Y	Y	N
UHF SCF	Y	N	N
HF DBOC	Y	N	N
CIS/RPA/TDHF	Y	N	N
TCSCF	Y	Y	N
CASSCF	Y	Y	N
RASSCF	Y	Y	N
RAS-CI	Y	N	N
RAS-CI DBOC	Y	N	N
RHF MP2	Y	Y	N
UHF/ROHF MP2	Y	N	N
RHF MP2-R12	Y	N	N
RHF/UHF/ROHF CCSD	Y	Y	N
RHF/UHF/ROHF CCSD(T)	Y	Y*	N
RHF/UHF/ROHF EOM-CCSD	Y	Y	N

* CCSD(T) gradients implemented only via an experimental code. A more efficient and robust implementation will appear in the next release.

1.4 Capabilities

This section needs to be updated with the new theory capabilities

PSI4 can perform *ab initio* computations employing basis sets of up to 32768 contracted Gaussian-type functions of virtually arbitrary orbital quantum number. PSI4 can recognize and exploit the largest Abelian subgroup of the point group describing the full symmetry of the molecule. Table 2 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 using analytic second derivatives, by finite differences of ana-

lytic gradients, or finite differences of energies. PSI4 can also compute an extensive list of one-electron properties.

1.5 Technical Support

The PSI4 package is distributed for free and without any guarantee of reliability, accuracy, 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.

2 Python

PSI4 has adopted the power of the Python scripting language to drive calculations. In doing so, users are allowed to perform complex calculations that would prove impossible with the old IPV1 style of input.

2.1 Preprocessor

At first glace at PSI4's new input style:

```
# cc-pVDZ H2O test single point
molecule {
o
h 1 0.957
h 1 0.957 2 104.5
}

set scf {
    basis cc-pVDZ
}

scf()
```

does not look like Python. This is because to provide the user a simpler format the input file is preprocessed into Python. Of course, if desired, you are free to put any Python command into your input file. In some cases you'll have to use Python (i.e. if you want to loop over a range of geometrical parameters or basis sets).

The above input file does the exact same thing that the input file provided in chapter 3's *Running a basic SCF calculation*. It provides a simple single-point energy computation on water.

```
# cc-pVDZ H2O test single point
molecule {
o
h 1 roh
h 1 roh 2 ahoh

roh = 0.957
ahoh = 104.5
}

set scf {
    basis cc-pVDZ
}

scf()
```

Both of these input files are automatically preprocessed to Python for you.

2.2 Preprocessor Keywords

molecule ***name*** **...**
Defines a molecule.

3 A PSI4 Tutorial

3.1 Before Getting Started: A Warning about Scratch Files

Generally, electronic structure programs like `PSI4` make significant use of disk drives. Therefore, 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 you will want to set up a default scratch path (as described in sections 4.5 and 4.6) because the `/tmp` directory is usually not large enough except for small test cases. In any event, you want to be very careful that you are not writing scratch files to an NFS-mounted directory that is physically attached to a fileserver elsewhere on the network.

3.2 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
```

3.2.1 IPV1-Style Input

PSI4 is a modular single executable, with each module performing specific tasks and computations. Which modules are run for a particular computation depends on the type of computation and the particular keywords specified in the input file. All keywords in PSI4 use the structure `keyword = value`, where values may be strings, booleans, integers, or real numbers. If the value is a string which contains a special character (such as a space or a dash) you must enclose the string in double quotation marks. You can give keywords in the input file for specific modules; however, in the first few examples, we will place all our keywords in one section of our input file called `psi`. Generally, every module you run during your computation will read the keywords in `psi`, so you can place all your keywords in this section if you choose to do so.

3.3 Running a basic SCF 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.

```
psi:(
 label = "cc-pVDZ SCF H2O"
 jobtype = sp
 wfn = scf
 reference = rhf
 basis = "cc-pVDZ"
 zmat = (
   o
   h 1 0.957
   h 1 0.957 2 104.5
  )
 )
```

In each computation, you can specify the type of wavefunction (keyword `wfn`), the reference wavefunction for post-Hartree-Fock computations (keyword `reference`), and the type of computation you want to perform (keyword `jobtype`). In the example above, we used a restricted Hartree-Fock (RHF) reference in an SCF computation of a single-point energy. To change the level of electron correlation, one would specify a different wavefunction type using the keyword `wfn`. In the example above, to perform an MP2 computation, simply set `wfn = mp2`.

3.4 Geometry Optimization and Vibrational Frequency Analysis

The above example was a simple single-point energy computation. To perform a different type of computation, change the keyword `jobtype`. In the example below, we will set up a CCSD geometry optimization. To illustrate a more flexible z-matrix input, we will now define variables for the bond length and bond angle (in the `zvars` section).

```
% 6-31G** H2O Test optimization calculation

psi: (
  label = "6-31G** SCF H2O"
  jobtype = opt
  wfn = ccsd
  reference = rhf
  dertype = first
  basis = "6-31G**"
  zmat = (
    o
    h 1 roh
    h 1 roh 2 ahoh
  )
  zvars = (
    roh     0.96031231
    ahoh  104.09437511
  )
)
```

Once you have optimized the geometry of a molecule, you might wish to perform a frequency analysis to determine the nature of the stationary point. To do this, change the value of `jobtype` to `freq`. For an SCF frequeny calculation, you would also set `dertype = second` to compute the second derivatives analytically. Unfortunately, analytical second derivitives are not available in `PSI4` for wavefunctions beyond SCF, so instead use the highest order analytical derivitives that are available for the type of wavefunction you have chosen. This information is given in Table 2. For our CCSD example, the highest-order derivitives available are first, so `dertype = first`.

```
% 6-31G** H2O Test computation of frequencies

psi: (
  label = "6-31G** SCF H2O"
  jobtype = freq
  wfn = ccsd
  reference = rhf
  dertype = first
```

```
  basis = "6-31G**"
  zmat = (
    o
    h 1 roh
    h 1 roh 2 ahoh
  )
  zvars = (
    roh     0.96031231
    ahoh  104.09437511
  )
)
```

3.5 More Advanced Input Options

If you wish to add comments to your input file, you can start any line with % and the line will be a comment line. This can make the input file easier to understand because you can provide explainations about each keyword. Another way to make the input file more organized is to seperate it into sections that correspond to particular modules the calculation will use. This can be particularly helpful for more complicated computations which can utilize many of keywords. In the example below, a CCSD(T) computation for the BH molecule is performed using a cc-pVDZ basis set. The keywords are divided into sections and several new keywords are introduced, including ones to specify symmetry and orbital occupations. Orbitial occupations are specified by a list of integers enclosed in parentheses. These integers give the number of orbitials which belong to each irreducible representation in the point group. The ordering of the irreps are those given by Cotton in *Chemical Applications of Group Theory*. In this example, comment lines will be included to explain the new keywords used.

```
psi: (
  wfn = ccsd_t
  reference = rhf
)

default: (
  label = "BH cc-pVDZ CCSD(T)"

% Allocating memory for the calculation
  memory = (600.0 MB)

% charge and multiplicity (2S+1) default to values of 0 and 1, respectively
  charge = 0
  multp = 1

% The program will generally guess the symmetry of the molecule, but
```

```
% it can be overridden.  Here we specify C2V because only D2H and its
% subgroups can be used by the program.
  symmetry = c2v

% Number of doubly-occupied orbitals per irrep can be specified manually
% if desired
  docc = (3 0 0 0)

% Freeze the 1A1 orbital (Boron 1s-like) in the CCSD(T) computation
  frozen_docc = (1 0 0 0)
)

% The input section contains information about the molecule and the basis
% set.  The geometry here is specified by cartesian coordinates.
input: (
  basis = "cc-pVDZ"
  units = angstroms
  geometry = (
    ( b      0.0000        0.0000        0.0000)
    ( h      0.0000        0.0000        0.8000)
      )
  origin = (0.0 0.0 0.0)
)
% The modular input structure lets you specify convergence criteria for
% each part of the computation separately
scf: (
  maxiter = 100
  convergence = 11
)
```

The final example of this tutorial demonstrates an example of a complete-active-space self-consistent-field (CASSCF) computation. CAS computations require specification of several additional keywords because you must specify which orbitals you wish to be in the active space. The notation and ordering for specifying CAS orbitals is the same as for occupied orbitals.

```
% 6-31G** H2O Test CASSCF Energy Point

psi: (
  label = "6-31G** CASSCF H2O"
  jobtype = sp
  wfn = casscf
  reference = rhf
% The restricted_docc orbitals are those which are optimized, but are not
```

```
% in the active space.
  restricted_docc = (1 0 0 0)

% The active space orbitals; here, the valence orbitals are chosen
  active          = (3 0 1 2)

  basis = "6-31G**"
  zmat = (
    o
    h 1 1.00
    h 1 1.00 2 103.1
  )
)
```

4 PSI3 Input Files

4.1 Syntax

PSI3 input files are case-insensitive and free-format, with a grammar designed for maximum flexibility and relative simplicity. Input values are assigned using the structure:

```
keyword = value
```

where `keyword` is the parameter chosen (e.g., `convergence`) and `value` has one of the following data types:

- string: A character sequence surrounded by double-quotes. Example: `basis = "cc-pVDZ"`
- integer: Any positive or negative number (or zero) with no decimal point. Example: `maxiter = 100`
- real: Any floating-point number. Example: `omega = 0.077357`
- boolean: `true`, `false`, `yes`, `no`, `1`, `0`.
- array: a parenthetical list of values of the above data types. Example: `docc = (3 0 1 1)`.

Note that the input parsing system is general enough to allow multidimensional arrays, with elements of more than one data type. A good example is the z-matrix keyword:

```
zmat = (
  (O)
  (H 1 r)
  (H 1 r 2 a)
)
```

For z-matrices, z-matrix variables, and Cartesian coordinates, it is also possible to discard the inner parentheses. The following is equivalent in this case:

```
zmat = (
  O
  H 1 r
  H 1 r 2 a
)
```

Keywords must grouped together in blocks, based on the module or modules that require them. The default block is labelled `psi:`, and most users will require only a `psi:` block when using `PSI3`. For example, the following is a simple input file for a single-point CCSD energy calculation on H_2O:

```
psi: (
  label = "6-31G**/CCSD H2O"
  wfn = ccsd
  reference = rhf
  jobtype = sp
  basis = "6-31G**"
  zmat = (
     O
     H 1 r
     H 1 r 2 a
  )
  zvars = (
     r 1.0
     a 104.5
  )
)
```

In this example, the `psi:` identifier collects all the keywords (of varying types) together. Every `PSI3` module will have access to every keyword in the `psi:` block by default. One may use other identifiers (e.g., `ccenergy:`) to separate certain keywords to be used only by selected modules. For example, consider the keyword `convergence`, which is used by several `PSI3` modules to determine the convergence criteria for constructing various types of wave functions. If one wanted to use a high convergence cutoff for the `PSI3` SCF module but a lower cutoff for the coupled cluster module, one could modify the above input:

```
psi: (
  ...
  convergence = 7
)
scf:convergence = 12
```

Note that, since we have only one keyword associated with the `scf:` block, we do not need to enclose it parentheses.

Some additional aspects of the `PSI3` grammar to keep in mind:

- The "%" character denotes a comment line, i.e. any information following the "%" up to the next linebreak is ignored by the program.
- Anything in between double quotes (i.e. strings) is case-sensitive.
- Multiple spaces are treated as a single space.

4.2 Specifying the Type of Computation

The most important keywords in a `PSI3` input file are those which tell the program what type of computation are to be performed. They `jobtype` keyword tells the `psi4` program whether this is a single-point computation, a geometry optimization, a vibrational frequency calculation, etc. The `reference` keyword specifies whether an RHF, ROHF, UHF, etc., reference is to be used for the SCF wavefunction. The `wfn` specifies what theoretical method is to be used, either SCF, determinant-based CI, coupled-cluster, etc. Also of critical importance are the charge and multiplicity of the molecule, the molecular geometry, and the basis set to be used. The latter two topics are discussed below in sections 4.3 and 4.7. General keywords determining the general type of computation to be performed are described below.

LABEL = string
This is a character string to be included in the output to help keep track of what computation has been run. It is not otherwise used by the program. There is no default.

JOBTYPE = string
This tells the program whether to run a single-point energy calculation (SP), a geometry optimization (OPT), a series of calculations at different displaced geometries (DISP), a frequency calculation (FREQ), frequencies only for symmetric vibrational modes (SYMM_FREQ), a Diagonal Born-Oppenheimer Correction (DBOC) energy computation, or certain response properties (RESPONSE). The default is SP.

WFN = string
This specifies the wavefunction type. Possible values are:
SCF, MP2, MP2R12, CIS, DETCI, CASSCF, RASSCF, CCSD, CCSD_T, BCCD, BCCD_T, EOM_CCSD, ZAPTN.

REFERENCE = string
This specifies the type of SCF calculation one wants to do. It can be one of RHF (for a closed shell singlet), ROHF (for a restricted open shell calculation), UHF (for an unrestricted open shell calculation), or TWOCON (for a two configuration singlet). The default is RHF.

MULTP = integer
Specifies the multiplicity of the molecule, i.e., 2S+1. Default is 1 (singlet).

CHARGE = integer
Specifies the charge of the molecule. Default is 0.

DERTYPE = string
This specifies the order of the derivative that is to be obtained. The default is NONE (energy only).

DOCC = integer vector
This gives the number of doubly occupied orbitals in each irreducible representation. There is no default. If this is not given, `cscf` will attempt to guess at the occupations.

SOCC = integer vector
This gives the number of singly occupied orbitals in each irreducible representation. There is no default. If this is not given, `cscf` will attempt to guess at the occupations.

FREEZE_CORE = string
`PSI3` can automatically freeze core orbitals. Core orbitals are defined as follows:

```
 H-Be  no core
 B-Ne  1s
Na-Ar  small: 1s2s
       large: 1s2s2p
```

YES or TRUE will freeze the core orbitals, SMALL or LARGE are for elements Na-Ar. The default is NO or FALSE. Always check to make sure that the occupations are correct!

4.3 Geometry Specification

The molecular geometry may be specified using either Cartesian a Z-matrix coordinates. Cartesian coordinates are specified via the keyword `geometry`:

```
geometry = (
   atomname1 x1 y1 z1
   atomname2 x2 y2 z2
   atomname3 x3 y3 z3
            ...
   atomnameN xN yN zN
)
```

where `atomname`*i* can take the following values:

- The element symbol: H, He, Li, Be, B, etc.

- The full element name: hydrogen, helium, lithium, etc.
- As a *ghost* atom with the symbol, G, or name, ghost. A ghost atom has a formal charge 0.0, and can be useful to specify the location of the off-nucleus basis functions.
- As a *dummy* atom with the symbol, X. Dummy atoms can be useful only to specify Z-matrix coordinates of proper symmetry or which contain linear fragments.

Hence the following two examples are equivalent to one another:

```
geometry = (
   H 0.0 0.0 0.0
   f 1.0 0.0 0.0
   Li 3.0 0.0 0.0
   BE 6.0 0.0 0.0
)

geometry = (
   hydrogen  0.0 0.0 0.0
   FLUORINE  1.0 0.0 0.0
   Lithium   3.0 0.0 0.0
   beryllium 6.0 0.0 0.0
)
```

It is also possible to include an inner set of parentheses around each line containing `atomname1 x1 y1 z1`.

The keyword `units` specifies the units for the coordinates:

- `units = angstrom` – angstroms (Å), default;
- `units = bohr` – atomic units (Bohr);

Z-matrix coordinates are specified using the keyword `zmat`:

```
zmat = (
   atomname1
   atomname2 ref21 bond_dist2
   atomname3 ref31 bond_dist3 ref32 bond_angle3
   atomname4 ref41 bond_dist4 ref42 bond_angle4 ref43 tors_angle4
   atomname5 ref51 bond_dist5 ref52 bond_angle5 ref53 tors_angle5
                              ...
   atomnameN refN1 bond_distN refN2 bond_angleN refN3 tors_angleN
)
```

where

- `bond_dist`i is the distance (in units specified by keyword `units`) from nucleus number i to nucleus number `ref`i`1`. The units
- `bond_angle`i is the angle formed by nuclei i, `ref`i`1`, and `ref`i`2`;
- `tors_angle`i is the torsion angle formed by nuclei i, `ref`i`1`, `ref`i`2`, and `ref`i`3`;

4.4 Molecular Symmetry

PSI3 can determine automatically the largest Abelian point group for a valid framework of centers (including ghost atoms, but dummy atoms are ignored). It will then use the symmetry properties of the system in computing the energy, forces, and other properties. However, in certain instances it is desirable to use less than the full symmetry of the molecule. The keyword `subgroup` is used to specify a subgroup of the full molecular point group. The allowed values are `c2v`, `c2h`, `d2`, `c2`, `cs`, `ci`, and `c1`. For certain combinations of a group and its subgroup there is no unique way to determine which subgroup is implied. For example, $D_{2\mathrm{h}}$ has 3 non-equivalent $C_{2\mathrm{v}}$ subgroups, e.g. $C_{2\mathrm{v}}(X)$ consists of symmetry operations $\hat{E}$, $\hat{C}_2(x)$, $\hat{\sigma}_{xy}$, and $\hat{\sigma_{xz}}$. To specify such subgroups precisely one has to use the keyword `unique_axis`. For example, the following input will specify the $C_{2\mathrm{v}}(X)$ subgroup of $D_{2\mathrm{h}}$ to be the computational point group:

```
psi: (
  ...
  geometry = (
       ...
  )
  units = angstrom
  subgroup = c2v
  unique_axis = x
)
```

Point Group	Cotton Ordering of Irreps
C_1	A
C_i	A_g A_u
C_2	A B
C_s	A' A"
C_{2h}	A_g B_g A_u B_u
C_{2v}	A_1 A_2 B_1 B_2
D_2	A B_1 B_2 B_3
D_{2h}	A_g B_{1g} B_{2g} B_{3g} A_u B_{1u} B_{2u} B_{3u}

4.5 Specifying Scratch Disk Usage in PSI3

Depending on the calculation, the PSI3 package often requires substantial temporary disk storage for integrals, wave function ampltiudes, etc. By default, PSI3 will write all such datafiles to /tmp (except for the checkpoint file, which is written to ./ by default). However, to allow for various customized arrangements of scratch disks, the PSI3 files: block gives the user considerable control over how temporary files are organized, including file names, scratch directories, and the ability to "stripe" files over several disks (much like RAID0 systems). This section of keywords is normally placed within the psi: section of input, but may be used for specific PSI3 modules, just like other keywords.

For example, if the user is working with PSI3 on a computer system with only one scratch disk (mounted at, e.g., /scr), one could identify the disk in the input file as follows:

```
psi: (
  ...
  files: (
    default: (
      nvolume = 1
      volume1 = "/scr/"
    )
  )
)
```

The nvolume keyword indicates the number of scratch directories/disks to be used to stripe files, and each of these is specified by a corresponding volumen keyword. (NB: the trailing slash "/" is essential in the directoy name.) Thus, in the above example, all temporary storage files generated by the various PSI3 modules would automatically be placed in the /scr directory.

By default, the scratch files are given the prefix "psi", and named "psi.nnn", where nnn is a number used by the PSI3 modules. The user can select a different prefix by specifying it in the input file with the name keyword:

```
psi: (
  ...
  files: (
    default: (
      name = "H2O"
      nvolume = 1
      volume1 = "/scr/"
    )
  )
)
```

The name keyword allows the user to store data associated with multiple calculations in the

same scratch area. Alternatively, one may specify the filename prefix on the command-line of the psi3 driver program (or any PSI3 module) with the -p argument:

```
psi3 -p H2O
```

If the user has multiple scratch areas available, PSI3 files may be automatically split (evenly) across them:

```
psi: (
  ...
  files: (
    default: (
      nvolume = 3
      volume1 = "/scr1/"
      volume2 = "/scr2/"
      volume3 = "/scr3/"
    )
  )
)
```

In this case, each PSI3 datafile will be written in chunks (65 kB each) to three separate files, e.g., /scr1/psi.72, /scr2/psi.72, and /scr3/psi.72. The maximum number of volumes allowed for striping files is eight (8), though this may be easily extended in the PSI3 I/O code, if necessary.

The format of the files section of input also allows the user to place selected files in alternative directories, such as the current working directory. This feature is especially important if some of the data need to be retained between calculations. For example, the following files: section will put file32 (the PSI3 checkpoint file) into the working directory, but all scratch files into the temporary areas:

```
psi: (
  ...
  files: (
    default: (
      nvolume = 3
      volume1 = "/scr1/"
      volume2 = "/scr2/"
      volume3 = "/scr3/"
    )
    file32: ( nvolume = 1  volume1 = "./" )
  )
)
```

4.6 The .psirc File

Users of PSI3 often find that they wish to use certain keywords or input sections in every calculation they run, especially those keywords associated with the `files:` section. The `.psirc` file, which is kept in the user's `$HOME` directory, helps to avoid repetition of keywords whose defaults are essentially user- or system-specific. A typical `.psirc` file would look like:

```
psi: (
  files: (
    default: (
      nvolume=3
      volume1 = "/tmp1/mylogin/"
      volume2 = "/tmp2/mylogin/"
      volume3 = "/tmp3/mylogin/"
    )
    file32: (nvolume=1 volume1 = "./")
  )
)
```

4.7 Specifying Basis Sets

PSI3 uses basis sets comprised of Cartesian or spherical harmonic Gaussian functions. A basis set is identified by a string, enclosed in double quotes. Currently, there exist three ways to specify which basis sets to use for which atoms:

- `basis = string` – all atoms use basis set type.
- `basis = (string1 string2 string3 ... stringN)` – `string` *i* specifies the basis set for atom *i*. Thus, the number of strings in the `basis` vector has to be the same as the number of atoms (including ghost atoms but excluding dummy atoms). Another restriction is that symmetry equivalent atoms should have same basis sets, otherwise `input` will use the string provided for the so-called unique atom out of the set of symmetry equivalent ones.
-
```
basis = (
  (element1 string1)
  (element2 string2)
          ...
  (elementN stringN)
)
```
  `string` *i* specifies the basis set for chemical element `element` *i*.

4.7.1 Default Basis Sets

PSI3 default basis sets are located in `pbasis.dat` which may be found by default in `$psipath/share`. Tables 3, 4, 5, and 6 list basis sets pre-defined in `pbasis.dat`.

The predefined basis sets use either spherical harmonics or Cartesian Gaussians, which is determined by the authors of the basis. Currently PSI3 cannot handle basis sets that consist of a mix of Cartesian and spherical harmonics Gaussians. Therefore there may be combinations of basis sets that are forbidden, e.g. `cc-pVTZ` and `6-31G**`. In such case one can override the predetermined choice of the type of the Gaussians by specifying the `puream` keyword. It takes two values, `true` or `false`, for spherical harmonics and Cartesian Gaussians, respectively.

4.7.2 Custom Basis Sets

If the basis set you desire is not already defined in PSI3, a custom set may be used by specifying its exponents and contraction coefficients (either in the input file or another file named `basis.dat`.) A contracted Cartesian Gaussian-type orbital

$$\phi_{\mathrm{CGTO}} = x^l y^m z^n \sum_i^N C_i \exp(-\alpha_i [x^2 + y^2 + z^2]) \tag{1}$$

where

$$L = l + m + n \tag{2}$$

is written as

```
basis: (
  ATOM_NAME: "BASIS_SET_LABEL" = (
    (L (C1  alpha1)
       (C2  alpha2)
       (C3  alpha3)
       ...
       (CN  alpha4))
    )
  )
```

One must further specify whether Cartesian or spherical harmonics Gaussians are to be used. One can specify that in two ways:

- It can be done on a basis by basis case, such as

```
basis: (
  "BASIS_SET_LABEL1":puream = true
  "BASIS_SET_LABEL2":puream = false
  "BASIS_SET_LABEL3":puream = true
  ....
)
```

Table 3: Pople-type basis sets available in PSI3

Basis Set	Atoms	Aliases
STO-3G	H-Ar	
3-21G	H-Ar	
6-31G	H-Ar, K, Ca, Cu	
6-31G*	H-Ar, K, Ca, Cu	6-31G(d)
6-31+G*	H-Ar	6-31+G(d)
6-31G**	H-Ar, K, Ca, Cu	6-31G(d,p)
6-311G	H-Ar	
6-311G*	H-Ar	6-311G(d)
6-311+G*	H-Ne	6-311+G(d)
6-311G**	H-Ar	6-311G(d,p)
6-311G(2df,2pd)	H-Ne	
6-311++G**	H, B-Ar	6-311++G(d,p)
6-311G(2d,2p)	H-Ar	
6-311++G(2d,2p)	H-Ar	
6-311++G(3df,3pd)	H-Ar	

Table 4: Huzinaga-Dunning basis sets available in PSI3

Basis Set	Atoms
(4S/2S)	H
(9S5P/4S2P)	B-F
(11S7P/6S4P)	Al-Cl
DZ	H, Li, B-F, Al-Cl
DZP	H, Li, Be, B-F, Na, Al-Cl
DZ-DIF	H, B-F, Al-Cl
DZP-DIF	H, B-F, Al-Cl
TZ2P	H, B-F, Al-Cl
TZ2PD	H
TZ2PF	H, B-F, Al-Cl
TZ-DIF	H, B-F, Al-Cl
TZ2P-DIF	H, B-F, Al-Cl
TZ2PD-DIF	H
TZ2PF-DIF	H, B-F, Al-Cl

Table 5: Wachters basis sets available in PSI3

Basis Set	Atoms
WACHTERS	K, Sc-Cu
WACHTERS-F	Sc-Cu

Table 6: Correlation-consistent basis sets available in PSI3

Basis Set **(N = D,T,Q,5,6)**	Atoms	Aliases
cc-pVNZ	H-Ar	CC-PVNZ
cc-pV(N+D)Z	Al-Ar	CC-PV(N+D)Z
cc-pCVNZ	B-Ne	CC-PCVNZ
aug-cc-pVNZ	H-He, B-Ne, Al-Ar	AUG-CC-PVNZ
aug-cc-pV(N+D)Z	Al-Ar	AUG-CC-PCV(N+d)Z
aug-cc-pCVNZ	B-F (N<6)	AUG-CC-PCVNZ
d-aug-cc-pVNZ	H	
pV7Z[1]	H, C, N, O, F, S	PV7Z
cc-pV7Z[2]	H, C, N, O, F, S	CC-PV7Z
aug-pV7Z[3]	H, C, N, O, F, S	AUG-PV7Z
aug-cc-pV7Z[4]	H, N, O, F	AUG-CC-PV7Z

By default, if `puream` is not given for a basis, then Cartesian Gaussians will be used.

- The choice between Cartesian or spherical harmonics Gaussian can be made globally by specifying `puream` keyword in the standard input section, e.g.

```
psi: (
  ...
  puream = true
  ...
)
```

Note that currently `PSI3` cannot handle basis sets that consist of a mix of Cartesian and spherical harmonics Gaussians.

Note that the basis set must be given in a separate `basis:` section of input, outside all other sections (including `psi:`). For example, the `PSI3` DZP basis set for carbon could be specified as:

```
basis: (
  carbon: "DZP" = (
    (S (   4232.6100      0.002029)
       (    634.8820      0.015535)
       (    146.0970      0.075411)
       (     42.4974      0.257121)
       (     14.1892      0.596555)
       (      1.9666      0.242517))
    (S (      5.1477      1.0))
    (S (      0.4962      1.0))
    (S (      0.1533      1.0))
    (P (     18.1557      0.018534)
       (      3.9864      0.115442)
       (      1.1429      0.386206)
       (      0.3594      0.640089))
    (P (      0.1146      1.0))
    (D (      0.75        1.0))
  )
)
```

Here are a couple of additional points that may be useful when specifying customized basis sets:

- Normally the `basis.dat` file is placed in the same directory as the main input file, but it may also be placed in a global location specified by the keyword `basisfile`:

```
basisfile = "/home/users/tool/chem/h2o/mybasis.in"
```

- To scale a basis set, a scale factor may be added as the last item in the specification of each contracted Gaussian function. For example, to scale the S functions in a 6-31G** basis for hydrogen, one would use the following

```
hydrogen: "6-31G**" =
    ( (S (    18.73113696     0.03349460)
         (     2.82539437     0.23472695)
         (     0.64012169     0.81375733) 1.2 )
      (S (     0.16127776     1.00000000) 1.2 )
      (P (     1.10000000     1.00000000))
     )
```

  In this example, both contracted S functions have their exponents scaled by a factor of $(1.2)^2 = 1.44$. The output file should show the exponents after scaling.

4.7.3 Automated Conversion of Basis Sets

The `PSI3` package is distributed with a Perl-based utility, named `g94_2_PSI3`, which will convert basis sets from the Gaussian ('94 or later) format to `PSI3` format automatically. This utility is especially useful for basis sets downloaded from the EMSL database at `http://www.emsl.pnl.gov/forms/basisform.html`. To use this utility, save the desired basis set to a file (e.g., `g94_basis.dat`) in the Gaussian format. Then execute:

```
g94_2_PSI3 < g94_basis.dat > basis.dat
```

You may either incorporate the results from the `basis.dat` file into your input file as described above, or place the results into a global `basis.dat` file. Be sure to surround the basis-set definition with the `basis:()` keyword (as shown in the above examples) or input parsing errors will result.

4.8 Specification of Ghost Atoms

To specify ghost atoms, use atom symbol `G` in `zmat` or `geometry` keywords:

```
zmat = (
  he
  g 1 r
)
basis = "aug-cc-pVTZ"
```

Basis sets for ghost atoms must be defined explicitly using `GHOST` as the element name:

```
basis:GHOST:"aug-cc-pVTZ": (
  ....
)
```

This method leads to replication of existing basis set definitions. It is usually more convenient to specify ghost atoms as regular atoms with zero charge:

```
zmat = (
  he
  he 1 r
)
charges = (2.0 0.0)
basis = "aug-cc-pVTZ"
```

In this example, the second helium atom is a "ghost" atom which carries helium's aug-cc-pVTZ basis set.

5 Theoretical Methods Available in PSI3

Several electronic structure methods are available in the PSI3 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. Less commonly used keywords are described in the man pages for each module.

5.1 Hartree-Fock Self-Consistent-Field

Hartree-Fock molecular orbital theory forms the cornerstone of ab initio quantum chemistry. Until the advances in the accuracy of Kohn-Sham density functional theory in the 1990's, Hartree-Fock theory was the method of choice for obtaining results for large molecules without resorting to standard empirical or semiempirical approaches. Molecular properties obtained by Hatree-Fock theory are generally at least qualitatively correct, although they can be quantitatively poor in many instances.

PSI3 solves the Hatree-Fock equations in a basis of Gaussian functions using an iterative, self-consistent-field (SCF) procedure. The final molecular orbitals are those which minimize the energy, subject to the electron configuration specified by the user (or guessed by the program). The process is continued until the largest change in an element of the density matrix drops below 10^{-n}, where n is an integer specified by the `convergence` keyword.

Of course the efficiency of the iterative procedure depends on the choice of initial guess. The `cscf` module will attempt to use previously obtained orbitals as a guess if they are available. This can be particularly advantageous when diffuse functions are present; in that case, it may be easiest to run the computation with a smaller basis and project those orbitals onto the larger basis by specifying the `--chkptmos` command-line argument or the `chkpt_mos=true` keyword in input when running the `input` program for the larger basis. If old MO's are not available, `cscf` uses a core Hamiltonian guess by default. The convergence of the SCF procedure is accelerated by Pulay's direct inversion of the iterative subspace (DIIS) approach, and it is possible to modify the behavior of the DIIS through various keywords, although this is seldom necessary.

It is important to point out that the SCF approach does not rigorously guarantee that the final orbitals actually correspond to a minimum in orbital space; at convergence, the only guarantee is that the gradient of the energy with respect to orbital rotations is zero: this could be a global minimum, a local minimum, or a saddle point in orbital rotation space. While this is not usually an issue (typically the lowest minimum consistent with the electron configuration is found), it can be a problem sometimes for radicals, diradicals, bond breaking, or unusual bonding situations. The `stable` module can be used to test for the stability of Hartree-Fock wave functions.

The most commonly used keywords are found below. More specialized keywords are available in the man pages.

MAXITER = integer
This gives the maximum number of iterations. The default is 40.

CONVERGENCE = integer
This specifies how tightly the wavefunction will be converged. Convergence is determined by comparing the RMS change in the density matrix ("delta P") to the given value. The convergence criterion is 10**(-integer). The default is 7 if both DERTYPE = NONE and WFN = SCF are given and 10 otherwise.

LEVELSHIFT = real
This specifies the level shift. The default is 1.

DIRECT = boolean
Specifies whether to do the SCF calculation with an integral-direct technique. The default is false.

NUM_THREADS = integer
Specified the number of threads to be used in the integral-direct computation (only valid if `DIRECT` is set to `true`). Default is 1.

PRINT_MOS = boolean
Specifies whether to print the molecular orbitals or not. The default is false.

5.2 Second-order Møller-Plesset Theory: MP2 and MP2-R12 methods

Second-order Møller-Plesset theory is one of the most basic wave function approaches which includes electron correlation directly. Due to its simplicity, the MP2 method is often the best level one can afford for a larger molecular system. At the other end of the spectrum, the MP2-R12 method of Kutzelnigg, Klopper, and co-workers is a promising approach to computing MP2 energies in the complete basis set limit for smaller systems. `PSI3` is one of the very few publicly available programs to feature a robust implementation of the MP2-R12 method.

PSI3 is capable of computing closed-shell MP2 and MP2-R12/A energies using integral-direct techniques and a multithreaded algorithm, which lends itself perfectly for execution on symmetric multiprocessor (SMP) machines. PSI3 is also capable of computing RHF, UHF, and ROHF (using semicanonical orbitals) MP2 energies and one-particle density matrices, and RHF MP2 analytic gradients. Occupied and virtual orbitals can be frozen during the energy calculation, but not for the calculation of the one-particle density matrix or the analytic gradient.

Table 7 summarizes these capabilities.

Reference	Method	Energy (conv)	Energy (integral-direct)	Gradient
RHF	MP2	Y	Y	Y
UHF	MP2	Y	N	N
ROHF	MP2	Y	N	N
RHF	MP2-R12/A	N	Y	N

Table 7: Current MP2 and MP2-R12 capabilities of PSI3.

5.2.1 Basic Keywords

To compute a ground-state MP2 or MP2-R12 energy at a fixed geometry, the following keywords are common:

WFN = string
Acceptable values are mp2 for MP2, mp2r12 [for MP2-R12/A] There is no default.

REFERENCE = string
The only acceptable value are rhf, uhf, and rohf. There is no default.

JOBTYPE = string
Acceptable values are sp and opt. There is no default.

MEMORY = (real MB)
Specified the amount of core memory to be used, in MB. Defaults to 256. Other units (e.g., KB or GB) are also allowed.

DIRECT = boolean
Specifies whether to use the conventional (false) or integral-direct (true) algorithm. Default is false.

NUM_THREADS = integer
Specified the number of threads to be used in the integral-direct computation (only valid if DIRECT is set to true). Default is 1.

FREEZE_CORE = boolean
Specifies whether core orbitals (which are determined automatically) are to be excluded from the correlated calculations. Default is `false`.

PRINT = integer
The desired print level for detailed output. Setting this to 2 is a good idea for larger calculations so that the progress of the calculation may be easily followed. Defaults to 0.

OPDM = boolean
If `true`, calculate the one-particle density matrix. The default is false.

OPDM_WRITE = boolean
If `true`, write the one-particle density matrix to disk.

OPDM_PRINT = boolean
If `true`, print the one-particle density matrix to the output file.

5.2.2 Using the MP2-R12 method

Although this manual is not a how-to on running quantum chemistry applications, the MP2-R12 method is a rather non-standard tool, hence a few comments on its use are appropriate.

1. The version of the MP2-R12 method implemented in `PSI3` is a so-called single-basis MP2-R12 method in standard approximation A. This means that a basis set rather complete in Hartree-Fock (or one-particle) sense is absolutely mandatory for meaningful computations with the MP2-R12 method. The user is strongly urged to read literature on linear R12 methods before using `PSI3` to compute MP2-R12 energies.

2. More robust, two-basis versions of the MP2-R12 method, also known as the auxiliary basis MP2-R12 method, have been implemented in a publicly available Massively Parallel Quantum Chemistry (MPQC) package (see `http://aros.ca.sandia.gov/~cljanss/mpqc/`). The two-basis version of the MP2-R12 method is a theoretically more sound approach, and thus should be preferred to the single-basis method. In some situations, however, it may make sense to use the single-basis method.

5.2.3 Larger Calculations

Here are a few recommendations for carrying out extended integral-direct MP2 and MP2-R12 calculations with `PSI3`:

1. While the integral-direct MP2 algorithm doesn't need any significant disk storage, the integral-direct algorithm for the MP2-R12 energy stores the transformed integrals to disk, hence very large computations will require a lot of disk space. In general the storage requirement is $16o^2N^2$ bytes, where o is the number of occupied orbitals, and N is the size of the basis.

2. If there is not enough memory to perform the computation in one pass, the program will do multiple passes through the entire set of integrals, hence your computation will run that many times longer. In such case, find the machine with the most memory and processors available.

3. On SMP machines, set the `NUM_THREADS` to the number of processors available for the job, or, if all processors are allocated for your job, set `NUM_THREADS` to *twice* the number of processors you have. Modern operating systems schedulers are usually very efficient at handling multithreaded programs, so the overhead of thread context switching is not significant, but using more threads may lead to better load balancing, and lower execution times. For example, on a 32-processor IBM eServer p690 we found that the optimal number of threads was 128. For the optimal performance, do a few runs with different number of threads and see which number works best. Avoid excessively large number of threads, as this descreases the net amount of memory available to the computation and thus may increase the number of passes.

4. Set the `MEMORY` keyword to the 90% of the available physical memory, at most. There is a small amount of overhead associated with the integral-direct algorithms that is not accounted for by the internal memory handling routines.

5. The implementation of the integral-direct MP2-R12 (and MP2) method in `PSI3` can run efficiently on SMP, or shared-memory, machines, by utilizing multiple processors via multithreaded approach. However, it cannot utilize distributed memory machines, such as commodity (PC) clusters and massively parallel machines, to their full potential, since one computation can only take advantage of one node of such machine at a time. In such environments, the aformentioned MPQC implementation of the MP2-R12 method should be preferred (see `http://aros.ca.sandia.gov/~cljanss/mpqc/`).

5.3 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. `PSI3` is capable of computing energies, analytic gradients, and linear response properties using a number of coupled cluster models. Table 8 summarizes these capabilities. This section describes how to carry out coupled cluster calculations within `PSI3`.

5.3.1 Basic Keywords

To compute a ground-state CCSD or CCSD(T) energy at a fixed geometry, the following keywords are common:

WFN = string
: Acceptable values are `ccsd`, `ccsd_t` [for CCSD(T)], `bccd` (for Brueckner-orbital-based CCD), or `bccd_t` [for Brueckner-orbital-based CCSD(T)] There is no default.

Table 8: Current coupled cluster capabilities of PSI3.

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)	N	N	–	–
UHF	CCSD	Y	Y	Y	N
UHF	CCSD(T)	Y	Y*	–	–
Brueckner	CCD	Y	N	N	N
Brueckner	CCD(T)	Y	N	–	–

*CCSD(T) gradients implemented via an experimental code. A more efficient and robust implementation will appear in the next release.

REFERENCE = string

Acceptable values are `reference = rhf`, `rohf`, or `uhf`. There is no default.

JOBTYPE = string

Acceptable values are `sp`, `opt`, `freq`, `oeprop`, or `response`. There is no default.

CONVERGENCE = integer

Sets the order of magnitude on the convergence of the CC wave function, perturbed wave function, and/or lambda parameters. The root-mean-square of the difference in amplitude vectors from consecutive iterations is used to determine the convergence. The default is 7.

MAXITER = integer

The maximum number of iterations allowed for solving the CC amplitude or lambda amplitude equations. Defaults to 50.

MEMORY = (real MB)

Specified the amount of core memory to be used, in MB. Defaults to 256. Other units (e.g., KB or GB) are also allowed.

BRUECKNER_CONV = integer

Specifies the order of magnitude convergence required for the Brueckner orbitals. The convergence is determined based on the largest T1 amplitude.

AO_BASIS = string

Specifies the algorithm to be used in computing the contribution of the four-virtual-index integrals ($\langle ab||cd\rangle$) to the CC amplitude equations. 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.

FREEZE_CORE = boolean

Specifies whether core orbitals (which are determined automatically) are to be excluded from the correlated calculations. Default is FALSE.

RESTART = boolean

Determine whether previous amplitude vectors may be used as guesses in a given CC calculation. Defaults to TRUE. 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.

PRINT = integer

The desired print level for detailed output. Setting this to 2 is a good idea for larger calculations so that the progress of the calculation may be easily followed. Defaults to 0.

CACHELEV = integer

Sets the level of automated cacheing of four-index quantities in the CC modules. These modules are capable of keeping in core as much as possible, various four-index quantities categorized by the number of virtual/unoccupied-orbital indices they contain. Setting CACHELEV=0 will cache nothing (wise and sometimes necessary for very large CC calculations), CACHELEV=1 will keep quantities with up to one virtual index in core (e.g., integrals of the form $\langle ij||ka\rangle$), CACHELEV=2 will keep quantities with up two two virtual indices in core (e.g., integrals of the form $\langle ij||ab\rangle$ or $\hat{T}_2$ amplitudes), CACHELEV=3 will keep three-virtual-index quantities in core, and CACHELEV=4 will keep everything in core. Note that the cache behavior is tempered by the MEMORY keyword, and items will be deleted from the cache (in an order determined based on the CACHETYEP keyword) as additional memory is required in a given calculation.

CACHETYPE = string

Specifies the type of cache to be used, either LOW or LRU. If CACHETYPE=LOW, then elements are deleted from the cache based on a predefined order of priority. If CACHETYPE=LRU, then elements are deleted from the cache based on a "least recently used" criterion: the least recently used item is the first to be deleted. The LOW criterion has been developed only ccenergy codes. The default is LRU for all CC modules except ccenergy.

NUM_AMPS = integer

Specifies the number of wave function amplitudes to print at the end of the energy calculation. Defaults to 10.

PRINT_MP2_AMPS = boolean
Specifies if the initial guess (MP2) amplitudes should be printed in the output file. Defaults to `FALSE`.

5.3.2 Larger Calculations

Here are a few recommendations for carrying out large-basis-set coupled cluster calculations with `PSI3`:

1. 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.

2. Set the `CACHELEV` 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.

5.3.3 Excited State Coupled Cluster Calculations

The most important keywords associated with EOM-CC calculations are:

STATES_PER_IRREP = (integer array)
Specifies the desired number of excited states per irreducible representation for both EOM-CC and CC-LR calculations. Note that the irreps in this keyword denote the final state symmetry, not the symmetry of the transition.

PRINT_SINGLES = boolean
Specifies whether information regarding the iterative solution to the single-excitation EOM-CC problem (normally used to obtain guesses for a ful EOM-CCSD calculation) will be printed.

RESIDUAL_TOL = integer
Specifies the order of magnitude cutoff used to determine the convergence of the Davidson algorithm residuals in the EOM-CC iterative procedure.

EVAL_TOL = integer
Specifies the order of magnitude cutoff used to determine the convergence of the final eigenvalues in the EOM-CC iterative procedure.

EOM_GUESS = (mixed array)
Specifies a set of single-excitation guess vectors for the EOM-CC procedure. This is especially useful for converging to difficult states. The `EOM_GUESS` keyword is an array,

each element of which includes an occupied orbital index (in coupled cluster ordering), a virtual orbital index, a weighting factor, and a spin (0 for α and 1 for β). The guess vector will be normalized after it is read, so only the relative magnitudes of the weight factors are important.

JOBTYPE = string
A value of `oeprop` will result in the calculation of oscillator strengths, rotational strengths, and dipole moments for an RHF reference and all but the rotational strengths for an ROHF or UHF reference.

5.3.4 Linear Response (CCLR) Calculations

The most important keywords associated with CC-LR calculations are:

JOBTYPE = string
A value of `RESPONSE` will invoke the linear-response programs.

PROPERTY = string
This leyword specifies the type or response property desired. Acceptable values are `POLARIZABILITY` (default) for dipole-polarizabilities and `ROTATION` for specific rotations.

OMEGA = real or (real UNITS) Specifies the desired frequency of the incident radiation field in CCLR calculations. Acceptable units are `HZ`, `NM`, and `EV`. If given without units, atomic units (Hartrees) are assumed.

MU_IRREPS = (integer array)
Specifies the irreducible representations associated with the x-, y-, and z-axes. This may be determined from the standard Cotton tables. Eventually this will be determined automatically by the program, so this keyword will go away.

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 correlations between electron motions. 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. The simplest standard CI method which improves upon Hartree-Fock is a CI which adds all singly and doubly substituted determinants (CISD). The CISD wavefunction has fallen out of favor because truncated CI wavefunctions short of full configuration interaction are not size-extensive, meaning that their quality degrades for larger molecules. MP2 offers a less expensive alternative whose quality does not degrade for larger molecules and which gives similar results to CISD for well-behaved molecules. CCSD is usually a more accurate alternative, at only slightly higher cost.

For the reasons stated above, the CI code in PSI3 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. The detci program is a fast, determinant-based CI program based upon the string formalism of Handy [1]. It can solve for restricted active space configuration interaction (RAS CI) wavefunctions as described by Olsen, Roos, Jorgensen, and Aa. Jensen [2]. 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.

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

As just mentioned, the PSI3 program 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 man pages for detci.

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 the detci program, 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., need to be in the psi:() or default:() sections of input so they may also be read by other modules.

5.4.1 Basic Keywords

WFN = string
Acceptable values for determinant-based CI computations in PSI3 are detci and, for CASSCF, detcas.

REFERENCE = string
Most reference types allowed by PSI3 are allowed by detci, except that uhf is not supported.

DERTYPE = string
Only single-point calculations are allowed for wfn = detci. For wfn = detcas, first derivatives are also available.

CONVERGENCE = integer

Convergence desired on the CI vector. Convergence is achieved when the RMS of the error in the CI vector is less than 10**(-n). The default is 4 for energies and 7 for gradients.

EX_LVL = integer

Excitation level for excitations into virtual orbitals (default 2, i.e. CISD). In a RAS CI, this is the number of electrons allowed in RAS III.

VAL_EX_LVL = integer

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_LVL + VAL_EX_LVL`. Defaults to zero.

FROZEN_DOCC = (integer array)

Core may be frozen by setting `FREEZE_CORE`. To manually select how many orbitals per irrep to freeze, use the `FROZEN_DOCC` keyword. Should be in `psi:()` or `default:()` sections of input. The number of lowest energy doubly occupied orbitals in each irreducible representation from which there will be no excitations. The Cotton ordering of the irredicible representations is used. The default is the zero vector.

FROZEN_UOCC = (integer array)

Should be in `psi:()` or `default:()` sections of input. The number of highest energy unoccupied orbitals in each irreducible representation into which there will be no excitations. The default is the zero vector.

RAS1 = (integer array)

Should be in `psi:()` or `default:()` sections of input. The number of orbitals for each irrep making up the RAS I space, from which a maximum of `EX_LVL + VAL_EX_LVL` excitations are allowed. This does not include frozen core orbitals. For a normal CI truncated at an excitation level such as CISD, CISDT, etc., it is not necessary to specify this or `RAS2` or `RAS3`. Note: this keyword must be visible to the `transqt` program also so that orbitals are ordered correctly (placing it in `default` or `psi` should be adequate).

RAS2 = (integer array)

Should be in `psi:()` or `default:()` sections of input. As above for `RAS1`, but for the RAS II subspace. No restrictions are placed on the occupancy of RAS II orbitals. Typically this will correspond to the conventional idea of an "active space" in multi-reference CI.

RAS3 = (integer array)

Should be in `psi:()` or `default:()` sections of input. As above for `RAS3`, but for the RAS III subspace. A maximum of `EX_LVL` electrons are allowed in RAS III.

MAXITER = integer

Maximum number of iterations to diagonalize the Hamiltonian. Defaults to 12.

NUM_ROOTS = integer

This value gives the number of roots which are to be obtained from the secular equations. The default is one. If more than one root is required, set `DIAG_METHOD` to `SEM` (or, for very small cases, `RSP` or `SEMTEST`). Note that only roots of the same irrep as the reference will be computed. To compute roots of a different irrep, one can use the `REF_SYM` keyword (for full CI only).

OPDM = boolean

If `TRUE`, compute the one-particle density matrix for each root. By default, it will be written to disk. Except for MCSCF computations (e.g., CASSCF, RASSCF), this will also turn on computation of dipole moments by default.

TRANSITION_DENSITY = boolean

If `TRUE`, compute the transition density matrix from the ground state to each other state obtained in the computation. By default, this information will be written to disk. Transition dipole moments will be evaluated in `detci`. 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 using the `subgroup` keyword in `psi:()` or `default:()` (see section 4.4).

DIPMOM = boolean

If `TRUE`, evaluate the dipole moment for each root (using the expectation value formula; orbital relaxation contributions are neglected). This is an alternative to evaluation using the `oeprop` module, which has more features.

REF_SYM = integer

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 is not supported for unit vector guesses.

MPN = boolean

If `TRUE`, compute MPn energies up to nth order, where MAXNVECT = n controls the maximum order energy computed. For open-shell systems (REF = ROHF, WFN = ZAPTN), ZAPTn energies are computed.

For larger computations, additional keywords may be required, as described in the `detci` man pages.

5.4.2 Arbitrary Order Perturbation Theory

`PSI3` is capable of computing arbitrary order Møller-Plesset perturbation theory (MPn, closed-shell systems) and Z-averaged perturbation theory (ZAPTn, open-shell systems) energies, invoked with MPN = TRUE. The maximum level of perturbation theory computed is controlled by MAXNVECT. Higher order energies (2n - 1 and 2n - 2) can be computed at no additional computational cost by using WIGNER = TRUE. By default, the nth order

energy is saved, but (2n - 1) or (2n - 2) order energies can be saved using SAVE_MPN2 = 1 or SAVE_MPN2 = 2, respectively.

For open-shell systems, arbitary order ZAPTn energies can be computed using WFN = ZAPTN and REF = ROHF. All other options are the same as closed-shell MPn.

5.5 Complete-Active-Space Self-Consistent-Field (CASSCF)

Multi-configurational self-consistent-field (MCSCF) is a general method for obtaining qualitatively correct wavefunctions for highly strained molecules, diradicals, or bond breaking reactions. The most commonly used MCSCF procedure is the complete-active-space self-consistent-field (CASSCF) approach [3], which includes all possible determinants (with the proper symmetry) that can be formed by distributing a set of active electrons among a set of active orbitals. The `detcasman` module performs CASSCF optimization of molecular orbitals via a two-step procedure in which the CI wavefunction is computed using `detci` and the orbital rotation step is computed using `detcas`. The `detcas` program is fairly simple and uses an approximate orbital Hessian [4] and a Newton-Raphson update, accelerated by Pulay's DIIS procedure [5]. We have also implemented a prototype version of the RASSCF method [6], which is another kind of MCSCF which is typically less complete (and less expensive) than CASSCF. However, orbital convergence for RASSCF can be difficult in our current implementation.

Inactive orbitals in the MCSCF may be specified by the `RESTRICTED_DOCC` and `RESTRICTED_UOCC` keywords. These orbitals will remain doubly-occupied or doubly-unoccupied, respectively, in the MCSCF wavefunction. However, the form of these orbitals will be optimized in the MCSCF procedure. It is also possible to literally freeze inactive orbitals in their original (SCF) form using the `FROZEN_DOCC` and `FROZEN_UOCC` keywords. This is not normally what one wishes to do in an MCSCF computation (e.g., it complicates the computation of gradients), but it can make the computations faster and is helpful in some circumstances where unphysical mixing of inactive and active occupied orbitals might occur. Presently, it is not possible to mix the use of restricted and frozen orbitals in `PSI3`.

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 the `detci` program, 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., need to be in the `psi:()` or `default:()` sections of input so they may also be read by other modules.

The ability to perform state-averaged [7, 8] CASSCF or RASSCF computations has been added. This is accomplished using the `average_states` keyword.

See the `casscf-sp` and `casscf-sa-sp` examples in the `tests` directory and the example below.

5.5.1 Basic Keywords

WFN = string
This may be `casscf` or `rasscf`.

REFERENCE = string
Any of the references allowed by `detci` should work (i.e., not `uhf`), but there should be no reason not to use `rhf`.

DERTYPE = string
At present, only energies (`none`) are supported; future releases will implement gradients (`first`).

CONVERGENCE = integer
Convergence desired on the orbital gradient. Convergence is achieved when the RMS of the error in the orbital gradient is less than 10**(-n). The default is 4 for energy calculations and 7 for gradients. Note that this is a different convergence criterion than for the `detci` program itself. These can be differentiated, if changed by the user, by placing the `CONVERGENCE` keywords within separate sections of input, such as `detcas: ( convergence = x )`.

ENERGY_CONVERGENCE = integer
Convergence desired on the total MCSCF energy. The default is 7.

RESTRICTED_DOCC = (integer array)
Should be in `psi:()` or `default:()` sections of input. The number of lowest energy doubly occupied orbitals in each irreducible representation from which there will be no excitations. These orbitals are optimized in the MCSCF. The Cotton ordering of the irredicible representations is used. The default is the zero vector.

RESTRICTED_UOCC = (integer array)
Should be in `psi:()` or `default:()` sections of input. The number of highest energy unoccupied orbitals in each irreducible representation into which there will be no excitations. These orbitals are optimized in the MCSCF. The default is the zero vector.

FROZEN_DOCC = (integer array)
Should be in `psi:()` or `default:()` sections of input. The number of lowest energy doubly occupied orbitals in each irreducible representation from which there will be no excitations. These orbitals are literally frozen and are not optimized in the MCSCF; usually one wishes to use `RESTRICTED_DOCC` instead. The current version of the program does not allow both `RESTRICTED_DOCC` and `FROZEN_DOCC`. Should be in `psi:()` or `default:()` sections of input. The Cotton ordering of the irredicible representations is used. The default is the zero vector.

FROZEN_UOCC = (integer array)
Should be in `psi:()` or `default:()` sections of input. The number of highest energy

unoccupied orbitals in each irreducible representation into which there will be no excitations. These orbitals are literally frozen and are not optimized in the MCSCF; usually one wishes to use `RESTRICTED_UOCC` instead. The current version of the program does not allow both `RESTRICTED_UOCC` and `FROZEN_UOCC`. Should be in `psi:()` or `default:()` sections of input. The default is the zero vector.

NCASITER = integer

Maximum number of iterations to optimize the orbitals. This option should be specified in the DEFAULT section of input, because it needs to be visible to the control program PSI. Defaults to 20.

AVERAGE_STATES = (integer array)

This gives a list of what states to average for the orbital optimization. States are numbered starting from 1.

PRINT = integer

This option determines the verbosity of the output. A value of 1 or 2 specifies minimal printing, a value of 3 specifies verbose printing. Values of 4 or 5 are used for debugging. Do not use level 5 unless the test case is very small (e.g. STO H_2O CISD).

5.5.2 Examples

```
% 6-31G** H2O Test CASSCF Energy Point

psi: (
  label = "6-31G** CASSCF H2O"
  jobtype = sp
  wfn = casscf
  reference = rhf
  restricted_docc = (1 0 0 0)
  active          = (3 0 1 2)
  basis = "6-31G**"
  zmat = (
    o
    h 1 1.00
    h 1 1.00 2 103.1
  )
)
```

Figure 1: Example of a CASSCF single-point calculation for H_2O using a valence active space $3a_1$ $1b_1$ $2b_2$.

6 Geometry Optimization

PSI3 is capable of carrying out geometry optimizations (minimization only, at present) for a variety of molecular structures using either analytic and numerical energy gradients.

When present, internal coordinates provided in the INTCO: section of the input will be read and used by PSI3. If these are missing, PSI3 will automatically generate and use redundant, simple internal coordinates for carrying out the optimization. These simple stretch, bend, torsion, and linear bend coordinates are determined by distance criteria using the input geometry.

By default, optimization is performed in redundant internal coordinates regardless of how the geometry was provided in the input. Alternatively, the user may specify zmat_simples=true, in which case the simple internal coordinates will be taken from the ZMAT given in the input file. Also, the user may specify optimization in non-redundant, delocalized internal coordinates with delocalize=true. In this case, the automatically generated simple coordinates are delocalized and redandancies are removed. Advanced users may wish to specify the simple internal coordinates in the intco.dat file, and then allow PSI3 to delocalize them.

Only those coordinates or combinations of coordinates that are specified by the "symm =" keyword in the INTCO: section are optimized. Coordinates can be approximately frozen by commenting them out within the "symm =" section. Geometrical constraints may be precisely imposed by the addition of a section with nearly the same format as in INTCO:. For example, to fix the distance between atoms 1 and 2, as well as the angle between atoms 2, 1 and 3 in an optimization, add the following to your input file.

```
fixed_intco: (
stre = (
(1 2)
)
bend = (
(2 1 3)
)
)
```

The constrained simple internals must be ones present (either manually or automatically) among the simple internals in the INTCO: section. Alternatively, the z-matrix input format may be used to specify constrained optimizations. If zmat_simples=true, then variables in the z-matrix which end in a dollar sign will be taken as simple internals to be optimized, and all other variables will be taken as simple internals to keep frozen.

To aid optimizations, force constants may be computed using "jobtype = symm_fc". The determined force constants will be saved in a binary file PSIF_OPTKING (currently file 1). Subsequent optimizations will read and use these force constants. In general, PSI3 looks for force constants in the following order: in this binary file, in the FCONST: section of the input, and in the fconst.dat file. If no force constants are found in any of these, then an empirical diagonal force constant matrix is generated.

For methods for which only energies are available, PSI3 will use non-redundant, symmetry-adapted delocalized internal coordinates to generate geometrical displacements for computing finite-difference gradients. The simple coordinates can be linearly combined by hand or automatically. The goal is to form 3N-6(5) symmetry-adapted internal coordinates. The automated delocalized coordinates may work for low-symmetry molecules without linear angles, but have not been extensively tested. For both analytic- and finite-difference-gradient optimization methods, Hessian updates are performed using the BFGS method.

The list below shows which coordinates are used by default for different types of jobs.
jobtype=freq dertype=first symmetry-adapted cartesians
jobtype=freq dertype=none symmetry-adapted cartesians
jobtype=fc dertype=first delocalized internals (or user-defined SALCs)
jobtype=symm_fc dertype=first delocalized internals (or user-defined SALCs)
jobtype=opt dertype=first redundant internals
jobtype=opt dertype=none delocalized internals (or user-defined SALCS)

The following keywords are pertinent for geometry optimizations.

JOBTYPE = string
This keyword must be set to OPT for geometry optimizations and SYMM_FC to compute force constants.

DERTYPE = string
This keyword must be set to NONE if only energies are available for the chosen method and FIRST if analytic gradients are available.

CONV = integer
The maximum force criteria for optimization is 10^{-conv}.

BFGS = boolean
If true (the default), a BFGS Hessian update is performed.

BFGS_USE_LAST = integer
This keyword is used to specify the number of gradient step for the BFGS update of the Hessian. The default is six.

SCALE_CONNECTIVITY = float
Determines how close atoms must be to be considered bonded in the automatic generation of the bonded list. The default is 1.3.

DELOCALIZE = integer
Whether to delocalize simple internal coordinates to attempt to produce a symmetry-adapted, non-redundant set.

MIX_TYPES = boolean
If set to false, different types of internal coordinates are not allowed to mix in the

formation of the delocalized coordinates. Although this produces cleaner coordinates, often the resulting delocalized coordinates form a redundant set.

ZMAT_SIMPLES = boolean
If set to true, the simple internal coordinates are taken from the zmat entry in the input file. The default is false.

POINTS = 3 or 5
Specifies a 3-point or a 5-point formula for optimization by energy points.

EDISP = float
The default displacment size (in au) for finite-difference computations. The default is 0.005.

FRAGMENT_DISTANCE_INVERSE = boolean
For interfragment coordinates. If true, then 1/R(AB) is used, if false, then R(AB) is used. The default is true.

FIX_INTRAFRAGMENT = boolean
If true, all intrafragment coordinates are constrained.

FIX_INTERFRAGMENT = boolean
If true, all interfragment coordinates are constrained.

DUMMY_AXIS_1 = 1 or 2 or 3
Specifies the axis for the location of a dummy atom for the definition of a linear bending coordinate. The default is 2.

DUMMY_AXIS_2 = 1 or 2 or 3
Specifies the axis for the location of a dummy atom for the definition of a linear bending coordinate. The default is 3.

TEST_B = boolean
If set to true, a numerical test of the B-matrix is performed.

PRINT_FCONST = boolean
If set to true and jobtype=symm_fc, then the force constants will be written to the fconst.dat file. This allows force constants to be reused even if the binary PSIF_OPTKING file is no longer present.

Print options
The following when set to true, print additional information to the output file: PRINT_SIMPLES, PRINT_PARAMS, PRINT_DELOCALIZE, PRINT_SYMMETRY, PRINT_HESSIAN, PRINT_CARTESIANS.

DISPLACEMENTS = ((integer float ...) ...)
A user may specify displacments along internal coordinates using this keyword. For example, displacements = ((2 0.01 3 0.01)) will compute a new cartesian geometry with the second and third internal coordinates increased by 0.01.

7 Vibrational Frequency Computations

PSI3 is also capable of computing harmonic vibrational frequencies for a number of different methods using energy points or analytic energy first or second derivatives. (At present, only RHF-SCF analytic second derivatives are available.) If analytic energy second derivatives are not available, PSI3 will generate displaced geometries along symmetry adapted cartesian coordinates, compute the appropriate energies or first derivatives, and use finite-difference methods to compute the Hessian.

The following keywords are pertinent for vibrational frequency analyses:

JOBTYPE = string
This keyword must be set to FREQ for frequency analyses.

DERTYPE = string
This keyword may be set to NONE if only energies are available for the chosen method, or FIRST if analytic gradients are available.

POINTS = 3 or 5
Specifies whether frequencies are determined by a 3-point or a 5-point formula of gradient differences. If only energy points are used, more displacements are required, but the effect of this keyword in terms of accuracy is the same.

Note: In some situations, vibrational frequency analysis via finite differences may fail if the full point group symmetry is specified via the symmetry *keyword. This happens because the user-given* symmetry *value can become incompatible with the actual symmetry of the molecule when energies or gradients are evaluated for symmetry-lowering displacements. In such situations, the user is advised to let the program determine the symmetry automatically, rather than specifying* symmetry *manually. Otherwise, an error such as the following may result:*

```
error: problem assigning number of operations per class
         *** stopping execution ***
```

The manual pages for the normco and intder95 modules contain information on additional tools useful in vibrational frequency analysis and coordinate transformation.

8 Evaluation of one-electron properties

PSI3 is capable of computing a number of one-electron properties Table 9 summarizes these capabilities. This section describes details of how to have PSI3 compute desired one-electron properties

Feature	On by default?	Notes
Electric dipole moment	Y	
Electric quadrupole moment	N	Set MPMAX to 2 or 3.
Electric octupole moment	N	Set MPMAX to 3.
Electrostatic potential	Y	At the nuclei; on 2-D grid set GRID=2.
Electric field	Y	At the nuclei.
Electric field gradient	Y	At the nuclei.
Hyperfine coupling constant	N	Set SPIN_PROP=true.
Relativistic (MVD) corrections	N	Set MPMAX to 2.
Electron density	Y	At the nuclei; on 2-D grid set GRID=2; on 3-D grid set GRID=6.
Spin density	N	Set SPIN_PROP=true; at the nuclei; on 3-D grid set GRID=6.
Electron density gradient	N	on 2-D grid set GRID=3.
Spin density gradient	N	on 2-D grid set GRID=3 and SPIN_PROP=true.
Electron density Laplacian	N	on 2-D grid set GRID=4.
Spin density Laplacian	N	on 2-D grid set GRID=4 and SPIN_PROP=true.
Molecular Orbitals (MO)	N	on 3-D grid set GRID=5.
Natural Orbitals (NO)	N	Set WRTNOS to true; written to file32.
MO/NO spatial extents	N	Set MPMAX to 2 or 3; MOs are used if WFN=SCF, otherwise NOs.

Table 9: Current one-electron property capabilities of PSI3.

8.1 Basic Keywords

To compute one-electron properties at a fixed geometry, the following keywords are common:

JOBTYPE = string
This keyword should be set to oeprop for PSI3 to compute electron properties. There is no default. For CI wavefunctions, limited properties such as dipole and transition moments may be evaluated directly in detci without having to specify JOBTYPE = oeprop.

WFN = string
Acceptable values are scf for HF, mp2 for MP2, detci for CI, detcas for CASSCF, and ccsd for CCSD. There is no default.

REFERENCE = string
Acceptable value are rhf and rohf. There is no default.

FREEZE_CORE = boolean
Specifies whether core orbitals (which are determined automatically) are to be excluded from the correlated calculations. Default is false.

PRINT = integer
The desired print level for detailed output. Defaults to 1.

MPMAX = integer
This integer specifies the highest-order electric multipole moment to be computed. Valid values are 1 (dipole), 2 (up to quadrupole), or 3 (up to octupole). Default is 1.

MP_REF = integer
This integer specifies the reference point for the evaluation of electric multipole moments. Valid values are 1 (center of mass), 2 (origin), 3 (center of electronic change) and 4 (center of the nuclear charge). For charge-neutral systems the choice of `MP_REF` is irrelevant. Default is 1.

GRID = integer
This integer specifies the type of one-electron property and the type of grid on which to evaluate it. The valid choices are

- `0` – compute nothing
- `1` – electrostatic potential on a 2-D grid
- `2` – electron density on a 2-D grid (spin density, if `SPIN_PROP=true`)
- `3` – projection of electron density gradient on a 2-D grid (spin density gradient, if `SPIN_PROP=true`)
- `4` – Laplacian of electron density on a 2-D grid (Laplacian of spin density, if `SPIN_PROP=true`)
- `5` – values of molecular orbitals on a 3-D grid
- `6` – electron density on a 3-D grid (spin density if `SPIN_PROP=true`)

Default is `0`.

NIX = integer
The number of grid points along the x direction. This parameter has be greater than 1. Default is 20.

NIY = integer
The number of grid points along the y direction. This parameter has be greater than 1. Default is 20.

NIZ = integer
The number of grid points along the z direction (if a 3-D grid is chosen). This parameter has be greater than 1. Default is 20.

GRID_FORMAT = string
This keyword specifies in which format to produce grid data. The only valid choice for 2-D grids is `plotmtv` (format of plotting software program `PlotMTV`). For 3-D grids, valid choices are `gausscube` (`Gaussian 94` cube format) and `megapovplus` (format of 3D rendering software `MegaPOV+`). The defaults are `plotmtv` and `gausscube` for 2-d and 3-d grids, respectively.

MO_TO_PLOT = vector

Specifies indices of the molecular orbitals to be computed on the 3-d grid. Indices can be specified as:

- unsigned integer - index in Pitzer ordering (ordered accoring to irreps, not eigenvalues). Ranges from 1 to the number of MOs.
- signed integer - index with respect to Fermi level. +1 means LUMO, +2 means second lowest virtual orbital, -1 means HOMO, etc.

All indices have to be either unsigned or signed, you can't mix and match, or you will get unpredictable results. Default is to compute HOMO and LUMO.

SPIN_PROP = boolean

Whether to compute spin-dependent properties. Default is `false`.

WRTNOS = boolean

If set to `true`, natural orbitals will be written to the checkpoint file. Default is `false`.

8.2 Evaluation of properties on rectalinear grids

`PSI3` can evaluate a number of one-electron properties on *rectalinear* 2-D and 3-D grids. In most cases, 3-D grids are utilized. In such cases you only need to specify the appropriate value for `GRID` and `PSI3` will automatically construct a rectalinear 3-D grid that covers the entire molecular system. However, there's no default way to construct a useful 2-D grid in general. Even in the 3-D case you may want to "zoom in" on a particular part of the molecule. Hence one needs to be able to specify general 2-D and 3-D grids. In the absence of graphical user interface, `PSI3` has a very flexible system for specifying arbitrary rectalinear grids.

The following keywords may be used in construction of the grid:

GRID_ORIGIN = real_vector

A vector of 3 real numbers, this keyword specifies the origin of the grid coordinate system. A rectangular grid box which envelops the entire molecule will be computed automatically if `GRID_ORIGIN` is missing, however, there is no default for 2-D grids.

GRID_UNIT_X = real_vector

A vector of 3 real numbers, this keyword specifies the direction of the first (x) side of the grid. It doesn't have have to be of unit length. There is no default for 2-D grids.

GRID_UNIT_Y = real_vector

A vector of 3 real numbers, this keyword specifies the direction of the second (y) side. It doesn't have to be of unit length or even orthogonal to `GRID_UNIT_X`. There is no default for 2-D grids.

GRID_UNIT_XY0 = real_vector
A vector of 2 real numbers, this keyword specifies the coordinates of the lower left corner of a 2-D grid in the 2-D coordinate system defined by `GRID_ORIGIN`, `GRID_UNIT_X`, and `GRID_UNIT_Y`. This keyword is only used to specify a 2-D grid. There is no default.

GRID_UNIT_XY1 = real_vector
A vector of 2 real numbers, this keyword specifies the coordinates of the upper right corner of a 2-D grid in the 2-D coordinate system defined by `GRID_ORIGIN`, `GRID_UNIT_X`, and `GRID_UNIT_Y`. This keyword is only used to specify a 2-D grid. There is no default.

GRID_UNIT_XYZ0 = real_vector
A vector of 3 real numbers, this keyword specifies the coordinates of the far lower left corner of a 3-D grid in the 3-D coordinate system defined by `GRID_ORIGIN`, `GRID_UNIT_X`, and `GRID_UNIT_Y`. This keyword is only used to specify a 3-D grid. There is no default.

GRID_UNIT_XYZ1 = real_vector
A vector of 3 real numbers, this keyword specifies the coordinates of the near upper right corner of a 3-D grid in the 3-D coordinate system defined by `GRID_ORIGIN`, `GRID_UNIT_X`, and `GRID_UNIT_Y`. This keyword is only used to specify a 3-D grid. There is no default.

In addition, the following keywords are useful for evaluation of certain properties on 2-D grids:

GRID_ZMIN = real
This keyword specifies the lower limit on displayed z-values for contour plots of electron density and its Laplacian. Only useful when `GRID=2` or `GRID=4`. Default is 0.0

GRID_ZMAX = real
This keyword specifies the upper limit on displayed z-values for contour plots of electron density and its Laplacian. Only useful when `GRID=2` or `GRID=4`. Default is 3.0

EDGRAD_LOGSCALE = integer
This keyword controls the logarithmic scaling of the produced electron density gradient plot. Turns the scaling off if set to zero, otherwise the higher value - the stronger the gradient field will be scaled. Recommended value (default) is 5. This keyword is only useful when `GRID=3`.

8.3 Grid specification mini-tutorial

Let's look at how to set up input for spin density evaluation on a two-dimensional grid. The relevant input section of `PSI3` might look like this:

```
jobtype = oeprop
```

```
grid = 2
spin_prop = true
grid_origin = (0.0 -5.0 -5.0)
grid_unit_x = (0.0 1.0 0.0)
grid_unit_y = (0.0 0.0 1.0)
grid_xy0 = (0.0 0.0)
grid_xy1 = (10.0 10.0)
nix = 30
niy = 30
```

`grid` specifies the type of a property and the type of a grid `oeprop` needs to compute. Since `spin_prop` is set and `grid=2`, the spin density will be evaluated on a grid.

Grid specification is a little bit tricky but very flexible. `grid_origin` specifies the origin of the rectangular coordinate system associated with the grid in the reference frame. `grid_unit_x` specifies a reference frame vector which designates the direction of the x-axis of the grid coordinate system. `grid_unit_y` is analogously a reference frame vector which, along with the `grid_unit_x`, completely specifies the grid coordinate system. `grid_unit_x` and `grid_unit_y` do not have to be normalized, neither they need to be orthogonal to either other - orthogonalization is done automatically to ensure that unit vectors of the grid coordinate system are normalized in the reference frame too. `grid_xy0` is a vector in the grid coordinate system that specifies a vertex of the grid rectangle with the most negative coordinates. Similarly, `grid_xy1` specifies a vertex of the the grid rectangle diagonally opposite to `grid_xy0`. Finally, `nix` and `niy` specify the number of intervals into which the x and y sides of the grid rectangle are subdivided. To summarize, the above input specifies a rectangular (in fact, square) 30 by 30 grid of dimensions 10.0 by 10.0 lying in the yz plane and centered at origin of molecular frame.

Running `PSI3` on such input will create a file called `sdens.dat` (for file names refer to man page on `oeprop`), which can be fed directly to `PlotMTV` to plot the 2-D data.

Specification of a three-dimensional grid for plotting MOs (`grid = 5`) or densities (`grid = 6`) is just slightly more complicated. For example, let's look at producing data for plotting a HOMO and a LUMO. The indices of the MOs which needs to be plotted will be specified by keyword `mo_to_plot`. The reference frame is specified by keywords `grid_origin`, `grid_unit_x` and `grid_origin_y` (the third axis of the grid coordinate system is specified by by the vector product of `grid_unit_x` and `grid_unit_y`). Since in this case we are dealing with the three-dimensional grid coordinate system, one needs to specify two diagonally opposite vertices of the grid box via `grid_xyz0` and `grid_xyz1`. The number of intervals along z is specified via `niz`. The relevant section of input file may look like this:

```
jobtype = oeprop

grid = 5
mo_to_plot = (-1 +1)
grid_origin = (-5.0 -5.0 -5.0)
```

```
grid_unit_x = (1.0 0.0 0.0)
grid_unit_y = (0.0 1.0 0.0)
grid_xyz0 = (0.0 0.0 0.0)
grid_xyz1 = (10.0 10.0 10.0)
nix = 30
niy = 30
niz = 30
```

Running PSI3 on input like this will produce a Gaussian Cube file called mo.cube, which can be used to render images of HOMO and LUMO using an external visualization software.

8.4 Plotting grid data

2-D grids should be plotted by an interactive visualization code PlotMTV. PlotMTV is a freeware code developed by Kenny Toh. It can be downloaded off many web sites in source or binary form.

3-D grids can be produced in two formats: megapovplus and gausscube (see GRID_FORMAT). First is used to render high-quality images with a program MegaPov (version 0.5). MegaPov is an unofficial patch for a ray-tracing code POV-Ray. Information on MegaPov can be found at http://nathan.kopp.com/patched.htm. Gaussian Cube files can be processed by a number of programs. We cannot recommend any particular program for that purpose here.

8.5 Visualizing Molecular Obitals with gOpenMol

The Gaussian Cube files generated by oeprop can be converted and viewed with gOpenMol. gOpenMol offers good looking plots in a graphical user interface. Information on downloading gOpenMol and samples of gOpenMol output may be found at http://www.csc.fi/gopenmol/.

Installation instructions are included with the gOpenMol download. Once installed, the first step to viewing molecular orbitals is to convert the mo.cube into a format that gOpenMol recognizes. Under the Run menu, select gCube2plt/g94cub2pl (cube) ..., this will bring up a window with the heading Run gCube2plt/g94cub2pl. In the input file name field, select the mo.cube file you want to convert. Likewise, in the output file name field type the name of the output file you want. Click the Apply button to perform the conversion. This procedure will create a .plt and a .crd file. Once converted, click Dismiss to close the window. The Gaussian Cube file is now converted and in a form that gOpenMol can recognize.

In order to view the molecular orbital, the first step is to import the coordinate file (.crd). This is done under the File menu→Import→Coords.... Again, a window will pop up. In the Import file name field chose the .crd you just created from the conversion procedure. Click apply, then Dismiss to close the window. Now we have to import the .plt file to view the molecular orbital. Under the the Plot menu selct Contour..., this will bring up a window.

In the File name field, either type the full path of the file name or use browse to select the `.plt` file you just created in the conversion, then click Import. In the Define contour levels we have to define the contour cutoffs for the positive and negative parts of the wave function seperately. I recommend trying 0.1 in the first box and -0.1 in the second. Click Apply to view the molecular orbital. You can change the colors of the positive and negative sections independently by clicking on the Colour button next to the respective cutoffs. Also, in the Details... section, you can fine tune the properties of the molecular orbital, such as, the opacity, solid vs. mesh, smoothness, and cullface state. You can play around with various settings to get the surface to look exactly how you want it to. There is more information in the Help→Tutorials menu on this subject as well as many other abilities of gOpenMol.

9 PSI3 Driver

The PSI3 suite of programs is built around a modular design. Any module can be run independently or the driver module, `psi3`, can parse the input file, recognize the calculation desired, and run all the necessary modules in the correct order. `psi3` reads the file `psi.dat` by default. `psi.dat` contains macros for several standard calculations, however, anything in `psi.dat` can be overriden by the user.

9.1 Environment Variables

PSIDATADIR

This flag is used to specify an alternate location for platform-independent read-only files such as `psi.dat` and `pbasis.dat`. By default, PSI3 will look for these files under `$psipath/share`.

9.2 Command-Line Options

-i or -f

This flag is used to specify the input file name, e.g. `psi3 -i h2o.in` where `h2o.in` is the name of the input file. By default, `psi3` and the other PSI3 modules look for `input.dat`.

-o

This flag is used to specify the output file name, e.g. `psi3 -o h2o.out` where `h2o.out` is the name of the output file. By default, `psi3` and the other PSI3 modules look for `output.dat`.

-p

This flag is used to specify the PSI3 file prefix, e.g. `psi3 -p h2o.dzp` where `h2o.dzp` is the prefix that will be used for all PSI3 files. By default, `psi3` and the other PSI3 modules use `psi` for the file prefix. Hence, the checkpoint file is by default called `psi.32`.

-n

This flag tells `psi3` not to run the `input` module. This flag is useful for scripting and debugging.

-c

This flag tells `psi3` to check the input and print out the list of programs which would be executed to STDOUT. Equivalent to `check = true` in the input file.

-m

This flag tells `psi3` not to run the cleanup module `psiclean`. Usually, `psiclean` is invoked by the `$done` macro in `psi.dat`. This flag is useful for scripting and debugging.

9.3 Input Format

The `psi3` module searches through the default keyword path (first `PSI` then `DEFAULT`) for the following keywords:

JOBTYPE = string

This keyword specifies what kind of calculation to run.

WFN = string

This keyword specifies the type of wave function.

REFERENCE = string

This keyword specifies the spin-reference.

DERTYPE = string

This keyword specifies the order of the derivative to be used. The default is `none`.

OPT = boolean

Set equal to `true` if performing a geometry optimization. The default is `false`.

CHECK = boolean

If `true`, `psi3` will parse your input file and print the sequence of programs to be executed to STDOUT. The default is `false`.

EXEC = string vector

The `EXEC` vector contains a list of commands to be executed by `psi3`. Explicit commands can be entered in double quotes, or preset variables can be entered using the convention $variable, e.g.

```
psi: (
  exec = ("ints")
)
```

or

```
psi: (
  ints = "ints"
  exec = ($ints)
)
```

9.4 Loop Control

Loop control is handled with the `repeat` and `end` commands. The syntax is:

```
repeat n [commands to be executed] end
```

where `n` is the number of times to repeat the loop. An inspection of the `psi.dat` file will show that this is how geometry optimizations and finite displacements are performed.

10 Additional Documentation

Additional information and the most recent version of this manual may be found at the `PSI3` website `www.psicode.org`. More complete information on the installation of the `PSI3` package is available in the PSI3 Installation Manual. For programmers, the PSI3 Programmer's Manual is available online along with complete library documentation.

A `PSI4` Reference

T. Daniel Crawford, C. David Sherrill, Edward F. Valeev, Justin T. Fermann, Rollin A. King, Matthew L. Leininger, Shawn T. Brown, Curtis L. Janssen, Edward T. Seidl, Joseph P. Kenny, and Wesley D. Allen, *J. Comput. Chem.* **28**, 1610-1616 (2007).

B Keywords Recognized by Each Module

B.1 CCDENSITY

AEL

Type: boolean **Default:** false

RELAX_OPDM

Type: boolean **Default:** false

AO_BASIS

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

GAUGE

Type: string **Default:** LENGTH

DERTYPE

Type: string **Default:** NONE

WFN

Type: string **Default:** SCF

STATES_PER_IRREP

Type: array **Default:** No Default

TOLERANCE

Type: integer **Default:** 14

CACHELEV

Type: integer **Default:** 2

PROP_ALL

Type: boolean **Default:** false

PROP_SYM

Type: integer **Default:** 0

REFERENCE

Type: string **Default:** RHF

CONNECT_XI

Type: boolean **Default:** false

PROP_ROOT

Type: integer **Default:** 0

B.2 CCENERGY

ABCD

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

DERTYPE

Possible Values: NONE, FIRST, RESPONSE
Type: string **Default:** NONE

SCS_CCSD

Type: boolean **Default:** 0

SCS_MP2

Type: boolean **Default:** 0

CC_SCALE_OS

Type: double **Default:** 1

PROPERTY

Possible Values: POLARIZABILITY, ROTATION, MAGNETIZABILITY, ROA, ALL
Type: string **Default:** POLARIZABILITY

DIIS

Type: boolean **Default:** true

FREEZE_CORE

The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

LOCAL_WEAKP

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

MAXITER

Type: integer **Default:** 50

ANALYZE

Type: boolean **Default:** 0

AO_BASIS

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

LOCAL_PAIRDEF

Possible Values: BP, RESPONSE
Type: string **Default:** BP

CONVERGENCE

Type: integer **Default:** 7

WFN

Possible Values: CCSD, CCSD_T, EOM_CCSD, LEOM_CCSD, BCCD, BCCD_T, CC2, CC3, EOM_CC2, EOM_CC3, CCSD_MVD
Type: string **Default:** NONE

RESTART

Type: boolean **Default:** 1

T3_WS_INCORE

Type: boolean **Default:** 0

NEWTRIPS

Type: boolean **Default:** 1

MP2_SCALE_SS

Type: double **Default:** 1

PRINT_PAIR_ENERGIES

Type: boolean **Default:** 0

BRUECKNER_CONV

Type: integer **Default:** 5

LOCAL_METHOD

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

NUM_AMPS

Type: integer **Default:** 10

SCSN_MP2

Type: boolean **Default:** 0

CC_SCALE_SS

Type: double **Default:** 1

LOCAL_MOS

Type: double **Default:** 0

CACHETYPE

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

LOCAL

Type: boolean **Default:** 0

LOCAL_CUTOFF

Type: double **Default:** 0

T2_COUPLED

Type: boolean **Default:** false

CACHELEV

Type: integer **Default:** 2

MP2_SCALE_OS

Type: double **Default:** 1

NTHREADS

Type: integer **Default:** 1

SPINADAPT_ENERGIES

Type: boolean **Default:** false

FORCE_RESTART

Type: boolean **Default:** 0

REFERENCE

Type: string **Default:** RHF

PRINT_MP2_AMPS

Type: boolean **Default:** 0

B.3 CCLAMBDA

ABCD

Type: string **Default:** NEW

DERTYPE

Type: string **Default:** NONE

LOCAL_METHOD

Type: string **Default:** WERNER

NUM_AMPS

Type: integer **Default:** 10

FREEZE_CORE

The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

MAXITER

Type: integer **Default:** 50

DIIS

Type: boolean **Default:** true

LOCAL_CUTOFF

Type: double **Default:** 0

LOCAL_WEAKP

Type: string **Default:** NONE

PROP_ALL

Type: boolean **Default:** false

LOCAL

Type: boolean **Default:** false

PROP_SYM

Type: integer **Default:** 1

AO_BASIS

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

SEKINO

Type: boolean **Default:** false

CONVERGENCE

Type: integer **Default:** 7

LOCAL_PAIRDEF

Type: string **Default:** No Default

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0

WFN

Type: string **Default:** SCF

CACHELEV

Type: integer **Default:** 2

PROP_ROOT

Type: integer **Default:** 1

RESTART

Type: boolean **Default:** false

STATES_PER_IRREP

Type: array **Default:** No Default

JOBTYPE

Type: string **Default:** No Default

LOCAL_FILTER_SINGLES

Type: boolean **Default:** true

B.4 RESPONSE

OMEGA

Type: array **Default:** No Default

PROPERTY

Type: string **Default:** POLARIZABILITY

REFERENCE

Type: string **Default:** RHF

B.5 DETCI

A_RAS3_MAX maximum number of alpha electrons in RAS III
Type: integer **Default:** -1

DERTYPE Derivative level
Possible Values: NONE, FIRST
Type: string **Default:** NONE

MPN_SCHMIDT
If TRUE, an orthonormal vector space is employed rather than storing the kth order wfn
Type: boolean **Default:** false

WIGNER
Use Wigner formulas in the Empn series?
Type: boolean **Default:** false

HD_AVE
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

MIXED4
This determines whether 'mixed' excitations involving RAS IV are allowed into the CI space. This is useful for placing additional constraints on a RAS CI.
Type: boolean **Default:** true

CC_VAL_EX_LVL
The CC valence excitation level
Type: integer **Default:** 0

EX_LVL
The CI excitation level
Type: integer **Default:** 2

NODFILE
If TRUE, 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

NPRINT
number of important determinants to print out
Type: integer **Default:** 20

RAS4_MAX
maximum number of electrons in RAS IV
Type: integer **Default:** -1

SAVE_MPN2
If TRUE, save MP(2n-1) energy; else, save MPn energy
Type: boolean **Default:** false

B_RAS3_MAX
maximum number of beta electrons in RAS III
Type: integer **Default:** -1

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
FCI_STRINGS	Store strings specifically for FCI? **Type:** boolean **Default:** false
HD_OTF	If TRUE the diagonal elements of the Hamiltonian matrix are computed on-the-fly, otherwise a diagonal element vector is written to a separate file on disk. **Type:** boolean **Default:** true
NUM_ROOTS	number of CI roots to find **Type:** integer **Default:** 1
REF_SYM	Open-shell type */ options.add_str("OPENTYPE", "NONE", "NONE HIGHSPIN SINGLET"); /*- irrep for CI vectors; -1 = find automatically **Type:** integer **Default:** -1
WFN	Wavefunction type **Possible Values:** DETCI, CI, ZAPTN, DETCAS, CASSCF, RASSCF **Type:** string **Default:** No Default
H0_BLOCK_COUPLING	Use coupling block in preconditioner? **Type:** boolean **Default:** false
R4S	Restrict strings with e- in RAS IV: i.e. if an electron is in RAS IV, then the holes in RAS I must equal the particles in RAS III + RAS IV else the string is discarded **Type:** boolean **Default:** false
CALC_SSQ	If TRUE, calculate the value of $< S^2 >$ for each root **Type:** boolean **Default:** false
CC_A_RAS3_MAX	maximum number of alpha electrons in RAS III, for CC **Type:** integer **Default:** -1
H0_GUESS_SIZE	size of H0 block for initial guess **Type:** integer **Default:** 100
PRINT_CIBLKS	print a summary of the CI blocks? **Type:** boolean **Default:** false
RAS34_MAX	maximum number of electrons in RAS III + IV **Type:** integer **Default:** -1
CC_NPRINT	number of important CC amps per ex lvl to print **Type:** integer **Default:** 10
H0_BLOCK_COUPLING_SIZE	size of coupling block in preconditioner **Type:** integer **Default:** 0

VAL_EX_LVL
The valence excitation level
Type: integer **Default:** 0

CC_B_RAS3_MAX
maximum number of beta electrons in RAS III, for CC
Type: integer **Default:** -1

H0_BLOCKSIZE
size of H0 block in preconditioner
Type: integer **Default:** 400

CC_EX_LVL
The CC excitation level
Type: integer **Default:** 2

CC_RAS4_MAX
maximum number of electrons in RAS IV, for CC
Type: integer **Default:** -1

MIXED
This determines whether 'mixed' RAS II/RAS III excitations are allowed into the CI space. This is useful for placing additional constraints on a RAS CI.
Type: boolean **Default:** true

REPL_OTF
Tells DETCI whether or not to do string replacements on the fly. 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

E_CONVERGE
E converge value
Type: integer **Default:** 10

ISTOP
stop after setting up CI space
Type: boolean **Default:** false

DETCI_FREEZE_CORE
Freeze core orbitals?
Type: boolean **Default:** true

D_CONVERGE
D converge value
Type: integer **Default:** 8

PERTURBATION_PARAMETER
z in H = H0 + z * H1
Type: double **Default:** 1

RAS3_MAX
maximum number of electrons in RAS III
Type: integer **Default:** -1

B.6 PSIMRCC

ACTIVE
Type: array **Default:** No Default

ACTIVE_DOCC

Type: array **Default:** No Default

ACTV

Type: array **Default:** No Default

CONVERGENCE

Type: integer **Default:** 9

CORR_ANSATZ

Possible Values: SR, MK, BW, APBW
Type: string **Default:** MK

CORR_WFN

Possible Values: PT2, CCSD, MP2-CCSD, CCSD_T
Type: string **Default:** CCSD

CORR_DOCC

Type: array **Default:** No Default

COUPLING_TERMS

Type: boolean **Default:** true

DAMPING_FACTOR

Type: integer **Default:** 0

PT_ENERGY

Possible Values: SECOND_ORDER, SCS_SECOND_ORDER, PSEUDO_SECOND_ORDER, SCS_PSEUDO_SECOND_ORDER
Type: string **Default:** SECOND_ORDER

FAVG_CCSD_T

Type: boolean **Default:** false

LOCK_SINGLET

Type: boolean **Default:** false

RESTRICTED_DOCC

Type: array **Default:** No Default

START_DIIS

Type: integer **Default:** 2

ZERO_INTERNAL_AMPS

Type: boolean **Default:** true

FROZEN_DOCC

Type: array **Default:** No Default

MAXITER

Type: integer **Default:** 100

OFFDIAGONAL_CCSD_T

Type: boolean **Default:** true

DEBUG

Type: integer **Default:** 0

DIAGONALIZE_HEFF

Type: boolean **Default:** false

TIKHONOW_MAX

Type: integer **Default:** 5

WFN

Possible Values: MRCCSD
Type: string **Default:** MRCCSD

DIIS_TRIPLES

Type: boolean **Default:** false

FROZEN_UOCC

Type: array **Default:** No Default

CORR_CCSD_T

Possible Values: STANDARD, PITTNER
Type: string **Default:** STANDARD

MAXDIIS

Type: integer **Default:** 7

CORR_ACTV

Type: array **Default:** No Default

CORR_FOCC

Type: array **Default:** No Default

CORR_FVIR

Type: array **Default:** No Default

CORR_REFERENCE

Possible Values: RHF, ROHF, TCSCF, MCSCF, GENERAL
Type: string **Default:** GENERAL

COUPLING

Possible Values: NONE, LINEAR, QUADRATIC, CUBIC
Type: string **Default:** CUBIC

DENOMINATOR_SHIFT

Type: integer **Default:** 0

NUM_THREADS

Type: integer **Default:** 1

PRINT_HEFF

Type: boolean **Default:** false

TIKHONOW_TRIPLES

Type: boolean **Default:** false

ONLY_CLOSED_SHELL

Type: boolean **Default:** false

PERT_CBS_COUPLING

Type: boolean **Default:** true

RESTRICTED_TRIPLES

Type: boolean **Default:** false

PERT_CBS

Type: boolean **Default:** false

TIKHONOW_OMEGA

Type: integer **Default:** 0

USE_SPIN_SYMMETRY

Type: boolean **Default:** true

DIAGONAL_CCSD_T

Type: boolean **Default:** true

TRIPLES_ALGORITHM

Possible Values: SPIN_ADAPTED, RESTRICTED, UNRESTRICTED
Type: string **Default:** RESTRICTED

HEFF4

Type: boolean **Default:** true

MP2_CCSD_METHOD

Possible Values: I, IA, II
Type: string **Default:** II

NEL

Type: integer **Default:** 0

USE_DIIS

Type: boolean **Default:** true

CORR_CHARGE

Type: integer **Default:** 0

WFN_SYM

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

MP2_GUESS

Type: boolean **Default:** true

ROOT

Type: integer **Default:** 1

B.7 STABLE

CACHELEV

Type: integer **Default:** 2

NUM_EVECS_PRINT

Type: integer **Default:** 0

REFERENCE

Type: string **Default:** RHF

SCALE

Type: double **Default:** 0

FOLLOW

Type: boolean **Default:** false

ROTATION_METHOD

Type: integer **Default:** 0

B.8 CUSP

FROZEN_DOCC

Type: array **Default:** No Default

FROZEN_UOCC

Type: array **Default:** No Default

B.9 DFMP2

BASIS

Basis, needed by Python
Type: string **Default:** NONE

DEBUG

Debugging information?
Type: integer **Default:** 0

RI_BASIS_MP2	RI Basis, needed by Python **Type:** string **Default:** No Default
RI_FITTING_TYPE	DFMP2 Fitting Type **Possible Values:** FINISHED, RAW, CHOLESKY **Type:** string **Default:** FINISHED
SCHWARZ_CUTOFF	Schwarz cutoff **Type:** double **Default:** 0
DFMP2_MEM_FACTOR	 **Type:** double **Default:** 0
FITTING_CONDITIONING	DFMP2 Fitting conditioning **Possible Values:** RAW, FINISHED **Type:** string **Default:** FINISHED
RI_INTS_NUM_THREADS	Number of threads to compute integrals with. 0 is wild card **Type:** integer **Default:** 0
RI_MAX_COND	Max condition number in auxiliary basis **Type:** double **Default:** 1
ROWS_PER_READ	-Maximum number of rows to read/write in each DF-MP2 operation **Type:** integer **Default:** 0
DFMP2_TYPE	DFMP2 Algorithm (usually for debugging) **Possible Values:** DEFAULT, DISK, CORE, OLD **Type:** string **Default:** DEFAULT
FITTING_SYMMETRY	DFMP2 Fitting symmetry **Possible Values:** SYMMETRIC, ASYMMETRIC **Type:** string **Default:** SYMMETRIC
FITTING_INVERSION	DFMP2 Fitting inversion **Possible Values:** EIG, CHOLESKY, SOLVE **Type:** string **Default:** EIG
PARALLEL_DFMP2	Parallel algoritmh? **Type:** boolean **Default:** false
D_CONVERGE	-Log10 of the density convergence criterion **Type:** integer **Default:** 8
E_CONVERGE	-Log10 of the energy convergence criterion **Type:** integer **Default:** 8
SCALE_SS	SS Scale **Type:** double **Default:** 1

B.10 CCSORT

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

DERTYPE
Type: string **Default:** NONE

EOM_REFERENCE
Type: string **Default:** RHF

KEEP_OEIFILE
Type: boolean **Default:** false

LOCAL_METHOD
Type: string **Default:** WERNER

LOCAL_DOMAIN_SEP
Type: boolean **Default:** false

PROPERTY
Type: string **Default:** POLARIZABILITY

FREEZE_CORE
The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

TOLERANCE
Type: integer **Default:** 14

LOCAL_CORE_CUTOFF
Type: double **Default:** 0

LOCAL_WEAKP
Type: string **Default:** NONE

LOCAL
Type: boolean **Default:** false

LOCAL_CUTOFF
Type: double **Default:** 0

OMEGA
Type: array **Default:** No Default

LOCAL_DOMAIN_MAG
Type: boolean **Default:** false

LOCAL_PAIRDEF
Type: string **Default:** BP

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0

WFN

Type: string **Default:** No Default

CACHELEV

Type: integer **Default:** 2

REFERENCE

Type: string **Default:** RHF

KEEP_TEIFILE

Type: boolean **Default:** false

LOCAL_DOMAIN_POLAR

Type: boolean **Default:** false

LOCAL_FILTER_SINGLES

Type: boolean **Default:** false

B.11 SCF

BASIS
primary basis set
Type: string **Default:** No Default

CARTESIAN_RESOLUTION_Y
Grid y resolution (npoints)
Type: integer **Default:** 1

CARTESIAN_RESOLUTION_Z
Grid z resolution (npoints)
Type: integer **Default:** 1

RI_SCF_SAVE
Should we make sure to save restart information for RI SCF?
Type: boolean **Default:** false

SAD_SCHWARZ_CUTOFF
SAD Guess Schwarz Sieve (for rough molecular F)
Type: double **Default:** 1E-7

DFT_RADIAL_POINTS
The number of radial points in the DFT grid
Type: integer **Default:** 99

DFT_SPHERICAL_TYPE
The spherical quadrature type for DFT, usually Lebedev
Possible Values: LEBEDEV, EM
Type: string **Default:** LEBEDEV

MAXITER
The maximum number of iterations
Type: integer **Default:** 100

DFT_VORONOI_A2
The far-field alpha within each cell for Voronoi-type point grouping

	Type: double	**Default:** 3
MULTP	$(2\times M\_s+1)$, e.g. 1 for a singlet state, 2 for a doublet, 3 for a triplet, etc.	
	Type: integer	**Default:** 0
LAMBDA	How big is the perturbation?	
	Type: double	**Default:** 0
SCF_TYPE	Tells which way to run SCF **Possible Values:** PK, OUT_OF_CORE, DIRECT, DF, PSEUDOSPECTRAL, POISSON, L_DF, CD, 1C_CD	
	Type: string	**Default:** PK
CARTESIAN_EXTENTS	Grid extents [xl, xr, yl, yr, zl, zr]	
	Type: array	**Default:** No Default
CARTESIAN_OVERAGE	Grid overage at sides	
	Type: double	**Default:** 2
CHARGE_CUTOFF	Atomic Charge cutoff (for primary domain)	
	Type: double	**Default:** 0
CHOLESKY_INTEGRALS_ONLY	Cholesky Integral factory only?	
	Type: boolean	**Default:** false
DFT_BASIS_EPSILON	The DFT basis cutoff	
	Type: double	**Default:** 0
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
S_MIN_EIGENVALUE	Minimum S matrix eigenvalue to be used before compensating for linear dependencies	
	Type: double	**Default:** 1E-7
DFT_GRID_NAME	The DFT grid specification, such as SG1 **Possible Values:** SG1	
	Type: string	**Default:** No Default
CARTESIAN_MO_INDICES	Grid MO indices	
	Type: array	**Default:** No Default
PERTURB_H	Whether to perturb the Hamiltonian or not	
	Type: boolean	**Default:** false
CARTESIAN_RESOLUTION	Grid global resolution (npoints)	
	Type: integer	**Default:** 30
CHARGE	The molecular charge	

	Type: integer	**Default:** 0
CHOLESKY_CUTOFF	Cholesky Cutoff **Type:** double	**Default:** 1E-4
N_CARTESIAN_MOS	Number MO indices **Type:** integer	**Default:** 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
DFT_BLOCK_SIZE	The number of grid points per evaluation block **Type:** integer	**Default:** 5000
SOCC	An array containing the number of singly-occupied orbitals per irrep (in Cotton order) **Type:** array	**Default:** No Default
STEPS_PER_LOCALIZE	Iterations per full Pipek-Mizey Localization **Type:** integer	**Default:** 1
DFT_FUNCTIONAL	The DFT combined functional name (for now) **Type:** string	**Default:** No Default
RI_SCF_STORAGE	The storage scheme for the three index tensors in density fitting **Possible Values:** DEFAULT, CORE, DISK **Type:** string	**Default:** DEFAULT
MOM_OCC	The absolute indices of orbitals to excite from in MOM (+/- for alpha/beta) **Type:** array	**Default:** No Default
RI_MAX_COND	Max J basis condition number to be allowed **Type:** double	**Default:** 1E8
RI_SCF_RESTART	Should we try to restart **Type:** boolean	**Default:** false
ROWS_PER_READ	Maximum number of rows to read/write in each DF-SCF operation **Type:** integer	**Default:** 0
POISSON_BASIS_SCF	The name of the poisson basis to be used in RI computations **Type:** string	**Default:** No Default
PRINT_MOS	Whether to print the molecular orbitals **Type:** boolean	**Default:** false
MOM_VIR	The absolute indices of orbitals to excite to in MOM (+/- for alpha/beta) **Type:** array	**Default:** No Default
R_EXT	Extended domain radius, Angstrom	

	Type: double	**Default:** 3
SAD_F_MIX_START	SAD Guess F-mix Iteration Start **Type:** integer	**Default:** 50
START_DIIS_ITER	The minimum iteration to start storing DIIS vectors **Type:** integer	**Default:** 1
CARTESIAN_FILENAME	Grid filename **Type:** string	**Default:** Grid.out
CARTESIAN_RESOLUTION_X	Grid x resolution (npoints) **Type:** integer	**Default:** 1
DFT_BOX_OVERAGE	The box overage for box-type point grouping **Type:** double	**Default:** 4
DFT_PRUNING_TYPE	The pruning scheme for the DFT grid **Possible Values:** NONE, AUTOMATIC **Type:** string	**Default:** NONE
RI_FITTING_TYPE	SCF Fitting Type **Possible Values:** FINISHED, RAW, CHOLESKY **Type:** string	**Default:** FINISHED
DFT_VORONOI_TYPE	The fuzzy Voronoi type for DFT, STRATMANN is best **Possible Values:** STRATMANN, BECKE **Type:** string	**Default:** STRATMANN
SAD_D_CONVERGE	SAD Guess Convergence in D **Type:** double	**Default:** 1E-5
E_CONVERGE	-Log10 of the energy convergence criterion **Type:** integer	**Default:** 8
RI_INTS_IO	IO caching for CP corrections, etc **Possible Values:** NONE, SAVE, LOAD **Type:** string	**Default:** NONE
SAD_CHOL_CUTOFF	SAD Guess Cholesky Cutoff (for eliminating redundancies) **Type:** double	**Default:** 1E-7
FREEZE_CORE	The scope of core orbitals to freeze in later correlated computations **Type:** string	**Default:** FALSE
RI_BASIS_SCF	The name of the auxiliary basis to be used in RI computations **Type:** string	**Default:** No Default
MAX_DIIS_VECTORS	The maximum number of error vectors stored for DIIS extrapolation **Type:** integer	**Default:** 10

SAD_MAXITER	SAD Guess Maxiter **Type:** integer **Default:** 50
DIIS	Whether DIIS extrapolation is used to accelerate convergence **Type:** boolean **Default:** true
DEBUG	The amount of debugging information to print **Type:** integer **Default:** false
DFT_VORONOI_A1	The near-field alpha within each cell for Voronoi-type point grouping **Type:** double **Default:** 1
DFT_GROUPING_TYPE	The point grouping scheme for the DFT grid **Possible Values:** BOXES, VORONOI **Type:** string **Default:** BOXES
SCHWARZ_CUTOFF	Minimum absolute TEI value for seive **Type:** double **Default:** 0
DFT_NUCLEAR_TYPE	The nuclear partition type for DFT, Treutler is best **Possible Values:** NAIVE, BECKE, TREUTLER **Type:** string **Default:** TREUTLER
OVERLAP_CUTOFF	Minimum absolute S matrix value for DF-SCF exchange **Type:** double **Default:** 0
GUESS	The guess type to be used in the computation **Possible Values:** CORE, GWH, SAD, READ **Type:** string **Default:** CORE
MOM_START	The iteration to start MOM on (or 0 for no MOM) **Type:** integer **Default:** 0
DOCC	An array containing the number of doubly-occupied orbitals per irrep (in Cotton order) **Type:** array **Default:** No Default
DFT_STRATMANN_ALPHA	The stratmann elliptical-confocal alpha cutoff in [0 1] **Type:** double **Default:** 0
SAD_E_CONVERGE	SAD Guess Convergence in E **Type:** double **Default:** 1E-5
SAD_PRINT	The amount of SAD information to print to the output **Type:** integer **Default:** 0
SAVE_CARTESIAN_GRID	Save a grid or not? **Type:** boolean **Default:** false
DFT_COORDINATE_TYPE	The Voronoi coordiante scheme for the DFT grid **Possible Values:** ELLIPTICAL, PROJECTION

Type: string **Default:** ELLIPTICAL

THREE_INDEX_CUTOFF
Minimum absolute three-index value for DF-SCF seive
Type: double **Default:** 0

MIN_DIIS_VECTORS
The minimum number of error vectors stored for DIIS extrapolation
Type: integer **Default:** 2

PARALLEL
Whether to run in parallel or not
Type: boolean **Default:** false

DFT_BOX_DELTA
The box width for box-type point grouping
Type: double **Default:** 4

DFT_SPHERICAL_POINTS
The number of spherical points in the DFT grid
Type: integer **Default:** 590

RI_INTS_NUM_THREADS
Max Number of threads for integrals (may be turned down if memory is an issue). 0 is blank
Type: integer **Default:** 0

S_ORTHOGONALIZATION
SO orthogonalization: symmetric or canonical?
Possible Values: SYMMETRIC, CANONICAL
Type: string **Default:** SYMMETRIC

FIND_RAW_J_COND
Should we find the raw fitting metric condition number?
Type: boolean **Default:** false

REFERENCE
The reference wavefunction used in the computation
Type: string **Default:** RHF

DFT_RADIAL_TYPE
The radial quadrature type for DFT, Treutler is best
Possible Values: TREUTLER, BECKE, EM, MURA, MULTI_EXP
Type: string **Default:** TREUTLER

D_CONVERGE
-Log10 of the density convergence criterion
Type: integer **Default:** 8

EXTERN
Look for an ExternalPotential object in Python or not?
Type: boolean **Default:** false

SAD_C
SAD Occupation Matrix Method
Possible Values: CHOLESKY, ID
Type: string **Default:** CHOLESKY

B.12 CCTRIPLES

DERTYPE
Type: string **Default:** NONE

NTHREADS

Type: integer **Default:** 1

REFERENCE

Type: string **Default:** RHF

WFN

Type: string **Default:** No Default

B.13 OEPROP

ASYMM_OPDM

Type: boolean **Default:** false

DELETE_ZVEC

Type: integer **Default:** 0

GRID_XY1

Type: array **Default:** No Default

LM_REF_XYZ

Type: array **Default:** No Default

NIY

Type: integer **Default:** 0

NIZ

Type: integer **Default:** 0

OPDM_FORMAT

Type: string **Default:** SQUARE

FREEZE_CORE

The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

MPMAX

Type: integer **Default:** 1

GRID_XYZ0

Type: array **Default:** No Default

GRID_ZMIN

Type: double **Default:** 0

NUM_ROOTS

Type: integer **Default:** 1

SPIN_PROP

Type: boolean **Default:** false

QED_DARWIN

Type: boolean **Default:** false

READ_OPDM

Type: boolean **Default:** true

WFN

Type: string **Default:** No Default

DOCC

Type: array **Default:** No Default

MP_REF_XYZ

Type: array **Default:** No Default

FINE_STRUCTURE_ALPHA

Type: double **Default:** 1

GRID_ORIGIN

Type: integer **Default:** 0

OPDM_FILE

Type: integer **Default:** 0

SOCC

Type: array **Default:** No Default

CORREL_CORR

Type: integer **Default:** 0

EDGRAD_LOGSCALE

Type: integer **Default:** 5

GRID

Type: integer **Default:** 0

GRID_FORMAT

Type: string **Default:** No Default

GRID_XYZ1

Type: array **Default:** No Default

GRID_ZMAX

Type: double **Default:** 3

NUC_ESP

Type: boolean **Default:** true

PRINT_NOS

Type: boolean **Default:** false

OPDM_BASIS

Possible Values: AO, MO
Type: string **Default:** MO

MO_TO_PLOT

Type: string **Default:** No Default

TRANSITION_DENSITY

Type: boolean **Default:** false

GRID_XY0

Type: array **Default:** No Default

NIX

Type: integer **Default:** 0

GRID_UNIT_X

Type: integer **Default:** 0

ROOT

Type: integer **Default:** 1

REFERENCE

Type: string **Default:** RHF

ZVEC_FILE

Type: double **Default:** 0

MP_REF

Type: integer **Default:** 0

WRTNOS

Type: boolean **Default:** false

B.14 DCFT

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

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

CHARGE — The molecular charge
Type: integer **Default:** 0

COMPUTE_TPDM	Whether to compute the full two particle density matrix at the end of the computation, for properties **Type:** boolean **Default:** 0
DAMPING_FACTOR	The damping factor used in the initial SCF procedure (0 - 1000) 0 means full standard SCF update is performed, 1000 will completely damp the iteration to the extent that no update is performed **Type:** integer **Default:** 0
DIIS_START	-log10 of the threshold below the RMS lambda and SCF error must be for DIIS to start **Type:** integer **Default:** 3
INT_THRESH	-log10 of the threshold below which an integral is considered to be zero **Type:** integer **Default:** 14
LAMBDA_MAXITER	The maximum number of lambda iterations per macro-iteration **Type:** integer **Default:** 50
MAX_DIIS	The maximum size of the DIIS subspace **Type:** integer **Default:** 6
DOCC	An array containing the number of singly-occupied orbitals per irrep (in Cotton order) **Type:** array **Default:** No Default
MEMORY	The amount of memory available (in Mb) **Type:** integer **Default:** 2000
MULTP	$(2\times M\_s+1)$, e.g. 1 for a singlet state, 2 for a doublet, 3 for a triplet, etc. **Type:** integer **Default:** 0
REGULARIZER	The shift applied to the denominator **Type:** double **Default:** 0
CONVERGENCE	The number of decimal digits required in the determination of lambda **Type:** integer **Default:** 10
LOCK_OCCUPATION	Whether to force the occupation to be that of the SCF starting point **Type:** boolean **Default:** true
SCF_CONV	The number of decimal digits required in the SCF density **Type:** integer **Default:** 8
DIIS_NUM_VECS	The number of DIIS vectors needed for extrapolation to start **Type:** integer **Default:** 3
IGNORE_TAU	Should the tau terms be included? **Type:** boolean **Default:** false
MAXITER	The maximum number iterations allowed

Type: integer **Default:** 40

RELAX_ORBITALS
Whether to relax the orbitals or not
Type: boolean **Default:** true

SCF_MAXITER
The maximum number of SCF iterations per cycle
Type: integer **Default:** 50

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

B.15 MCSCF

ACTIVE
The number of active orbitals, per irrep
Type: array **Default:** No Default

ACTV
The number of active orbitals, per irrep (alternative name for ACTIVE)
Type: array **Default:** No Default

CANONICALIZE_ACTIVE_FAVG
Type: boolean **Default:** false

CANONICALIZE_INACTIVE_FAVG
Type: boolean **Default:** false

ROTATE_MO_IRREP
Type: integer **Default:** 1

MAXITER
Type: integer **Default:** 100

TURN_ON_ACTV
Type: integer **Default:** 0

ROTATE_MO_P
Type: integer **Default:** 1

LEVELSHIFT
Type: integer **Default:** 0

MULTP
$(2\times M\_s + 1)$, e.g. 1 for a singlet state, 2 for a doublet, 3 for a triplet, etc.
Type: integer **Default:** 1

CI_DIIS
Type: boolean **Default:** false

CONVERGENCE
Type: integer **Default:** 9

DEBUG
The amount of debugging information to print
Type: integer **Default:** false

FORCE_TWOCON
Type: boolean **Default:** false

INTERNAL_ROTATIONS
Type: boolean **Default:** true

ROOT
Type: integer **Default:** 1

ROTATE_MO_Q
Type: integer **Default:** 2

WFN_SYM
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

ROTATE_MO_ANGLE
Type: integer **Default:** 0

USE_DIIS
Type: boolean **Default:** true

CHARGE
The molecular charge
Type: integer **Default:** 0

REFERENCE
Possible Values: RHF, ROHF, UHF, TWOCON, MCSCF, GENERAL
Type: string **Default:** RHF

DOCC
The number of doubly occupied orbitals, per irrep
Type: array **Default:** No Default

E_CONVERGE
-Log10 of the energy convergence criterion
Type: integer **Default:** 12

D_CONVERGE
-Log10 of the density convergence criterion
Type: integer **Default:** 12

NDIIS
Type: integer **Default:** 7

SOCC
The number of singly occupied orbitals, per irrep
Type: array **Default:** No Default

START_FAVG
Type: integer **Default:** 5

READ_MOS

Type: boolean **Default:** true

USE_FAVG

Type: boolean **Default:** false

B.16 MINTS

BASIS

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

B.17 SAPT

AIO_CPHF

Use asynchronous I/O in the CPHF solver
Type: boolean **Default:** false

DEBUG

The ubiquitous debug flag
Type: integer **Default:** 0

MAXITER

Max CPHF iterations
Type: integer **Default:** 50

OCC_CUTOFF

Natural Orbital Occupation Cutoff
Type: double **Default:** 1

SAPT_LEVEL

The level of theory for SAPT
Possible Values: SAPT0, MP2C
Type: string **Default:** SAPT0

SAPT_MEM_SAFETY

Memory safety
Type: double **Default:** 0

AIO_DFINTS

Use asynchronous I/O in the DF integral formation
Type: boolean **Default:** false

DENOMINATOR_DELTA

Maximum denominator error allowed (Max error norm in Delta tensor)
Type: double **Default:** 1

NAT_ORBS_T2

Use Natural Orbitals for T2's
Type: boolean **Default:** false

SCHWARZ_CUTOFF

Schwarz cutoff
Type: double **Default:** 1

NAT_ORBS

Compute Natural Orbitals
Type: boolean **Default:** false

NFRZ_B

Frozen Occupieds of Monomer B

	Type: integer	**Default:** 0
DIISVECS	DIIS vecs **Type:** integer	**Default:** 5
D_CONVERGE	D converge value **Type:** integer	**Default:** 8
E_CONVERGE	E converge value **Type:** integer	**Default:** 10
OMEGA_POINTS	Number of Omega points for Casimir-Polder **Type:** integer	**Default:** 8
RI_BASIS_SAPT	SAPT DF Basis **Type:** string	**Default:** No Default
OMEGA_CENTER	Omega (atomic wavenumbers) to center Casimir-Polder on **Type:** double	**Default:** 0
PRINT	The ubiquitous print flag **Type:** integer	**Default:** 1
NO_RESPONSE	Don't solve the CPHF equations **Type:** boolean	**Default:** false
NFRZ_A	Frozen Occupieds of Monomer A **Type:** integer	**Default:** 0

B.18 OPTKING

CONV_MAX_DE	QCHEM optimization criteria: maximum energy change **Type:** double	**Default:** 1
CONV_MAX_DISP	QCHEM optimization criteria: maximum displacement **Type:** double	**Default:** 1
CONV_MAX_FORCE	QCHEM optimization criteria: maximum force **Type:** double	**Default:** 3
FRAGMENT_MODE	Whether to treat multiple molecule fragments as a single bonded molecule; or via interfragment coordinates ; a primary difference is that in MULTI mode, the interfragment coordinates are not redundant. SINGLE, MULTI **Possible Values:** SINGLE, MULTI **Type:** string	**Default:** SINGLE
FREEZE_INTRAFRAGMENT	Whether to freeze all fragments rigid **Type:** boolean	**Default:** false
GENERATE_INTCOS_ONLY	Whether to only generate the internal coordinates and then stop true, false	

	Type: boolean **Default:** false
H_UPDATE	Choose from supported Hessian updates NONE, BFGS, MS, POWELL, BOFILL **Possible Values:** NONE, BFGS, MS, POWELL, BOFILL **Type:** string **Default:** BFGS
H_UPDATE_LIMIT_MAX	If the above is true, changes to the Hessian from the update are limited to the larger of (H_update_limit_scale)*(the previous value) and H_update_limit_max (in au). **Type:** double **Default:** 1
H_UPDATE_USE_LAST	How many previous steps' data to use in Hessian update; 0=use them all ; integer **Type:** integer **Default:** 6
INTRAFRAGMENT_STEP_LIMIT	Maximum step size in bohr or radian along an internal coordinate double **Type:** double **Default:** 0
RFO_FOLLOW_ROOT	Whether to 'follow' the initial RFO vector after the first step true, false **Type:** boolean **Default:** false
H_UPDATE_LIMIT	Whether to limit the magnitutde of changes caused by the Hessian update true, false **Type:** boolean **Default:** true
SCALE_CONNECTIVITY	When determining connectivity, a bond is assigned if interatomic distance is less than (this number) * sum of covalent radii double **Type:** double **Default:** 1
INTERFRAGMENT_H	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
READ_CARTESIAN_H	Read Cartesian Hessian **Type:** boolean **Default:** false
TEST_B	Whether to test B matrix **Type:** boolean **Default:** false
H_UPDATE_LIMIT_SCALE	If the above is true, changes to the Hessian from the update are limited to the larger of (H_update_limit_scale)*(the previous value) and H_update_limit_max (in au). **Type:** double **Default:** 0
TEST_DERIVATIVE_B	Whether to test derivative B matrix **Type:** boolean **Default:** false
INTERFRAGMENT_DISTANCE_INVERSE	Whether to use 1/R(AB) for stretching coordinate between fragments (or just R(AB))

Type: boolean **Default:** false

STEP_TYPE
Whether to do an ordinary Newton-Raphson step or an RFO step; allowed values = NR, RFO
Possible Values: RFO, NR
Type: string **Default:** RFO

RFO_ROOT
Which RFO root to follow; 0 indicates lowest (to a minimum); integer
Type: integer **Default:** 0

WRITE_FINAL_STEP_GEOMETRY
By default, optking prints and saves the last (previous) geometry at the end of an optimization, i.e., the one at which a gradient was computed. If this keyword is set to true, then the structure obtained from the last projected step is printed out and saved instead.
Type: boolean **Default:** false

INTERFRAGMENT_MODE
whether to use fixed linear combinations of atoms as reference points for interfragment coordinates or whether to use principal axes FIXED, PRINCIPAL_AXES
Possible Values: FIXED, INTERFRAGMENT
Type: string **Default:** FIXED

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

MAXIMUM_H_BOND_DISTANCE
For now, this is a general maximum distance for the definition of H-bonds
Type: double **Default:** 4

B.19 TRANSQT

AA_M_FILE
Type: integer **Default:** PSIF_MO_AA_TEI

AB_M_FILE
Type: integer **Default:** PSIF_MO_AB_TEI

DELETE_TPDM
Type: boolean **Default:** true

IVO
Type: boolean **Default:** false

SORTED_TEI_FILE
Type: integer **Default:** PSIF_MO_TEI

FZC_A_FILE
Type: integer **Default:** PSIF_OEI

KEEP_PRESORT

Type: boolean **Default:** false

PRINT_REORDER

Type: boolean **Default:** false

REORDER

Type: boolean **Default:** false

DERTYPE

Type: string **Default:** NONE

LAGRAN_HALVE

Type: boolean **Default:** false

MAX_BUCKETS

Type: integer **Default:** 499

PRINT_TE_INTEGRALS

Type: boolean **Default:** false

FREEZE_CORE

The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

OEI_FILE

Type: integer **Default:** PSIF_OEI

AO_BASIS

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

MODE

Possible Values: TO_MO, TO_AO
Type: string **Default:** TO_MO

PRINT_SORTED_OE_INTS

Type: boolean **Default:** false

WFN

Type: string **Default:** CCSD

FZC_B_FILE

Type: integer **Default:** PSIF_OEI

TPDM_FILE

Type: integer **Default:** PSIF_MO_TPDM

FZC_FILE

Type: integer **Default:** PSIF_OEI

OPDM_OUT_FILE

Type: integer **Default:** PSIF_AO_OPDM

DOCC

Type: array **Default:** No Default

MOORDER

Type: array **Default:** No Default

OEI_A_FILE

Type: integer **Default:** PSIF_OEI

SOCC

Type: array **Default:** No Default

FIRST_TMP_FILE

Type: integer **Default:** 150

KEEP_J

Type: boolean **Default:** false

DO_ALL_TEI

Type: boolean **Default:** false

MP2R12A

Possible Values: MP2R12AERI, MP2R12AR12, MP2R12AR12T1
Type: string **Default:** MP2R12AERI

LAGRAN_DOUBLE

Type: boolean **Default:** false

M_FILE

Type: integer **Default:** 0

J_FILE

Type: integer **Default:** 91

PRINT_MOS

Type: boolean **Default:** false

PSIMRCC

Type: boolean **Default:** false

SO_TEI_FILE

Type: integer **Default:** PSIF_SO_TEI

TOLERANCE

Type: integer **Default:** 14

SO_S_FILE

Type: integer **Default:** PSIF_OEI

SO_T_FILE

Type: integer **Default:** PSIF_OEI

CHECK_C_ORTHONORM

Type: boolean **Default:** false

SO_V_FILE

Type: integer **Default:** PSIF_OEI

BB_M_FILE

Type: integer **Default:** PSIF_MO_BB_TEI

DELETE_AO

Type: boolean **Default:** true

DELETE_RESTR_DOCC

Type: boolean **Default:** true

LAG_IN_FILE

Type: integer **Default:** PSIF_MO_LAG

OEI_B_FILE

Type: integer **Default:** PSIF_OEI

OPDM_IN_FILE

Type: integer **Default:** PSIF_MO_OPDM

PRESORT_FILE

Type: integer **Default:** PSIF_SO_PRESORT

PRINT_LVL

Type: integer **Default:** 1

PRINT_OE_INTEGRALS

Type: boolean **Default:** false

TPDM_ADD_REF

Type: boolean **Default:** false

PITZER

Type: boolean **Default:** false

PRINT_SORTED_TE_INTS

Type: boolean **Default:** false

QRHF

Type: boolean **Default:** false

REFERENCE

Type: string **Default:** RHF

B.20 TRANSQT2

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

PRINT_TEI
Type: boolean **Default:** false

DERTYPE
Type: string **Default:** NONE

WFN
Type: string **Default:** No Default

CACHELEV
Type: integer **Default:** 2

TOLERANCE
Type: integer **Default:** 14

DELETE_TEI
Type: boolean **Default:** true

REFERENCE
Type: string **Default:** RHF

B.21 CCHBAR

CACHELEV
Type: integer **Default:** 2

EOM_REFERENCE
Type: string **Default:** RHF

TAMPLITUDE
Type: boolean **Default:** false

WABEI_LOWDISK
Type: boolean **Default:** false

DERTYPE
Type: string **Default:** ENERGY

WFN
Type: string **Default:** SCF

B.22 LMP2

DIISSTART

Type: integer **Default:** 3

SCALE_OS

Type: double **Default:** 6

NEGLECT_DP

Type: boolean **Default:** 1

SCS_N

Type: boolean **Default:** false

DISTANT_PAIR

Type: double **Default:** 8

RI_LMP2

Type: boolean **Default:** false

ENERGY_CONV

Type: integer **Default:** 7

MAXITER

Type: integer **Default:** 50

FSKIP

Type: integer **Default:** 2

LOCAL_CUTOFF

Type: double **Default:** 0

RI_BASIS_MP2

The wavefunction desired

Type: string **Default:** No Default

SCS

Type: boolean **Default:** false

MEMORY

Type: integer **Default:** 2000

NDIIS

Type: integer **Default:** 6

REFERENCE

Possible Values: RHF

Type: string **Default:** RHF

RMS_CONV

Type: integer **Default:** 5

SCHWARTZ_TOL
Type: integer **Default:** 12

SCREEN_INTS
Type: boolean **Default:** false

WFN
Type: string **Default:** LMP2

USE_DIIS
Type: boolean **Default:** 1

SCALE_SS
Type: double **Default:** 1

SCREENING
Type: integer **Default:** 7

B.23 CCRESPONSE

ABCD
Type: string **Default:** NEW

GAUGE
Type: string **Default:** LENGTH

LOCAL_METHOD
Type: string **Default:** WERNER

NUM_AMPS
Type: integer **Default:** 5

DERTYPE
Type: string **Default:** NONE

PROPERTY
Type: string **Default:** POLARIZABILITY

DIIS
Type: boolean **Default:** 1

FREEZE_CORE
The scope of core orbitals to freeze in later correlated computations
Type: string **Default:** FALSE

LOCAL
Type: boolean **Default:** 0

MAXITER
Type: integer **Default:** 50

LOCAL_CUTOFF

Type: double **Default:** 0

LOCAL_WEAKP

Type: string **Default:** NONE

ANALYZE

Type: boolean **Default:** 0

CACHELEV

Type: integer **Default:** 2

CONVERGENCE

Type: integer **Default:** 7

LOCAL_FILER_SINGLES

Type: boolean **Default:** false

LOCAL_PAIRDEF

Type: string **Default:** NONE

OMEGA

Type: array **Default:** No Default

SEKINO

Type: boolean **Default:** 0

LOCAL_CPHF_CUTOFF

Type: double **Default:** 0

WFN

Type: string **Default:** SCF

RESTART

Type: boolean **Default:** 1

LINEAR

Type: boolean **Default:** 0

REFERENCE

Type: string **Default:** RHF

B.24 MP2

CACHELEV

Type: integer **Default:** 2

CACHETYPE

Possible Values: LRU, LOW

Type: string **Default:** LRU

REFERENECE

Type: string **Default:** RHF

JOBTYPE

Type: string **Default:** SP

SCALE_OS

Type: double **Default:** 6

SCALE_SS

Type: double **Default:** 1

SCS

Type: boolean **Default:** false

WFN

Type: string **Default:** No Default

DERTYPE

Type: string **Default:** NONE

SCS_N

Type: boolean **Default:** false

B.25 CIS

CONVERGENCE

Type: integer **Default:** 7

DOMAINS

Type: array **Default:** No Default

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

Type: integer **Default:** 500

STATES_PER_IRREP

Type: array **Default:** No Default

WFN

Possible Values: CCSD, CCSD_T, EOM_CCSD, CIS
Type: string **Default:** CIS

DIAG_METHOD

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

DOMAIN_PRINT

Type: boolean **Default:** 0

LOCAL_CUTOFF

Type: double **Default:** 0

REFERENCE

Possible Values: RHF, ROHF, UHF
Type: string **Default:** RHF

LOCAL

Type: boolean **Default:** false

LOCAL_AMP_PRINT_CUTOFF

Type: double **Default:** 0

B.26 MVO

CANONICAL

Type: boolean **Default:** false

DELETE_RESTR_DOCC

Type: boolean **Default:** true

FOCK_COEFF

Type: double **Default:** 0

FROZEN_UOCC

Type: array **Default:** No Default

FZC_FILE

Type: integer **Default:** PSIF_OEI

FZC_FOCK_COEFF

Type: double **Default:** 1

PRINT_MOS

Type: boolean **Default:** false

RESTRICTED_DOCC

Type: array **Default:** No Default

IVO

Type: boolean **Default:** false

RESTRICTED_UOCC

Type: array **Default:** No Default

FROZEN_DOCC

Type: array **Default:** No Default

OEI_ERASE

Type: boolean **Default:** false

FZC

Type: boolean **Default:** true

WFN

Type: string **Default:** CCSD

UNOS

Type: boolean **Default:** false

DOCC

Type: array **Default:** No Default

MP2NOS

Type: boolean **Default:** false

SOCC

Type: array **Default:** No Default

DOCC_VIRT

Type: array **Default:** No Default

B.27 CLAG

DERTYPE

Type: string **Default:** NONE

ROOT

Type: integer **Default:** 1

WFN

Type: string **Default:** NONE

WRITE_CAS_FILES

Type: boolean **Default:** 0

B.28 DFCC

BASIS
MO basis
Type: string **Default:** NONE

MAX_DIIS_VECS
Maximum DIIS vectors
Type: integer **Default:** 6

SCALE_OS
OS Scale
Type: double **Default:** 6

T_CONVERGE
Convergence of cluster amplitudes (RMS change)
Type: integer **Default:** 8

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

MAXITER
The maximum number iterations allowed
Type: integer **Default:** 40

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

RPA_ALPHA
RPA alpha parameter
Type: double **Default:** 1

DEBUG
Debugging information?
Type: integer **Default:** 0

DIIS
Turn on DIIS
Type: boolean **Default:** true

MP2_ALGORITHM
MP2 Algorithm:

Algorithm Keyword	MP2J	MP2K
DF	DF	DF
SOS	DF	-
MOS	DF(Omega)	-
PS	DF	PS
PS2	DF	PS/PS
PS3	PS	PS/PS

Possible Values: DF, SOS, MOS, PS, PS2, PS3
Type: string **Default:** DF

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

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

RI_BASIS_CC
DF basis for MO integrals

	Type: string	**Default:** NONE
DEALIAS_BASIS_CC	Dealias basis for PS integrals **Type:** string	**Default:** NONE
DENOMINATOR_DELTA	Maximum denominator error allowed (Max error norm in Delta tensor) **Type:** double	**Default:** 1
RPA_ALGORITHM	RPA algorithm: DF $\mathcal{O}(N^5)$, CD $\mathcal{O}(N^4)$ **Possible Values:** CD, DF **Type:** string	**Default:** CD
SCHWARZ_CUTOFF	Schwarz cutoff **Type:** double	**Default:** 0
FITTING_CONDITION	Desired Fitting condition (inverse of max condition number) **Type:** double	**Default:** 1
MIN_DIIS_VECS	Minimum DIIS vectors **Type:** integer	**Default:** 2
E_CONVERGE	Convergence of CC energy **Type:** integer	**Default:** 8
PS_GRID_FILE	Filename to read grid from **Type:** string	**Default:** No Default
SCALE_SS	SS Scale **Type:** double	**Default:** 1
WAVEFUNCTION	Type of wavefunction **Possible Values:** MP2, MP3, CCD, DRPA **Type:** string	**Default:** MP2
RPA_DELTA	RPA Cholesky delta **Type:** double	**Default:** 1
RPA_PLUS_EPSILON	Continue RPA numerical SPD Tolerance **Type:** double	**Default:** 1

C Expert Keywords Recognized by Each Module, for Advanced Users

C.1 DFMP2

SCALE_OS	OS Scale **Type:** double	**Default:** 6

D 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:-

basis_extrap	Various basis set extrapolation tests
cc1	RHF-CCSD 6-31G** all-electron optimization of the H2O molecule
cc2	6-31G** H2O Test CCSD energy, with Z-Matrix input
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.
cc5	RHF CCSD(T) aug-cc-pvtz frozen-core energy of C4NH4 Anion
cc5a	RHF CCSD(T) STO-3G frozen-core energy of C4NH4 Anion
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.
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.
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.
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.
props1	RHF STO-3G dipole moment computation, performed by applying a finite electric field and numerical differentiation.

References

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

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

[3] B. O. Roos, P. R. Taylor, and P. E. M. Siegbahn, Chem. Phys. **48**, 157 (1980).

[4] G. Chaban, M. W. Schmidt, and M. S. Gordon, Theor. Chem. Acc. **97**, 88 (1997).

[5] P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).

[6] P.-Å. Malmqvist, A. Rendell, and B. O. Roos, J. Phys. Chem. **94**, 5477 (1990).

[7] K. K. Docken and J. Hinze, J. Chem. Phys. **57**, 4928 (1972).

[8] K. Ruedenberg, L. M. Cheung, and S. T. Elbert, Int. J. Quantum Chem. **16**, 1069 (1979).