The PSI3 User's Manual

C. David Sherrill[a], T. Daniel Crawford[b], Edward F. Valeev[a],
Micah L. Abrams[a], and Rollin A. King[c]

[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*

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

PSI3 Version: 3.2.1 (stable)
Created on: April 23, 2004

Contents

1 Introduction

1.1 Overview

This manual explains how to use the PSI3 suite of *ab initio* quantum chemical programs. In this section, we provide an overview of some of the features of PSI3 along with the prerequisite steps for running calculations. Section 2 provides a brief tutorial to help new users get started. Section 3 offers further details into the structure of PSI3 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 PSI3 (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 PSI3 input and basis set files. For the latest PSI3 documentation, check www.psicode.org.

The PSI3 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 PSI3 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, PSI 3.2, 2003.

1.2 Obtaining and Installing PSI3

The latest version of the PSI3 program package may be obtained at www.psicode.org. The source code is available as a gzipped tar archive (named, for example, psi3.X.tar.gz), and binaries may be available for certain architectures. For detailed installation and testing instructions, please refer to the the PSI3 Installation Manual, available as part of the package or at the PSI3 website above.

1.3 Supported Architectures

The majority of PSI3 was developed on IBM RS/6000/AIX and x86/GNU Linux workstations. The complete list of tested architectures to which PSI3 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 PSI3 on your system if you have the prerequisite tools and math and utility libraries described in the installation manual.

Table 1: Platforms on which PSI3 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, AMD x86-64	32 and 64-bit
Apple OS X (Darwin) on PowerPC	
SGI IRIX64 (>6.5.15)	64-bit

Table 2: Summary of theoretical methods available in PSI3.

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
RAS-CI	Y	N	N
RAS-CI DBOC	Y	N	N
RHF MP2	Y	N	N
RHF MP2-R12	Y	N	N
RHF CCSD	Y	Y	N
ROHF CCSD	Y	Y	N
UHF CCSD	Y	Y	N
RHF CCSD(T)	Y	N	N
ROHF CCSD(T)	N	N	N
UHF CCSD(T)	Y	N	N
RHF EOM-CCSD	Y	N	N
ROHF EOM-CCSD	Y	N	N

1.4 Capabilities

PSI3 can perform *ab initio* computations employing basis sets of up to 32768 contracted Gaussian-type functions of virtually arbitrary orbital quantum number. PSI3 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 PSI3. 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 or by finite differences of analytic

gradients. PSI3 can also compute an extensive list of one-electron properties.

1.5 Technical Support

The PSI3 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.

2 A PSI3 Tutorial

Before going into all the details of the PSI3 input format, we will begin with a tutorial on how to run some very simple computations.

The first thing to point out is that electronic structure programs like PSI3 make significant use of disk drives. Therefore, for any but the smallest calculations, *it is very important to be sure that* PSI3 *is writing its temporary files to a disk drive physically attached to the computer running the computation.* If a user's directory is remotely mounted by NFS, and if PSI3 tries to write its temporary files to this directory, it will slow the program and the network down dramatically. To avoid this situation, by default PSI3 will write temporary files to /tmp. This will work for the small examples in this tutorial, but for real computations you will want to set up a default scratch file path as described in sections 3.5 and 3.6, since your /tmp directory may not be large enough.

The PSI3 suite of codes is built around a modular design which allows it great power and flexibility. Any module can be run independently (provided suitable datafiles, of course). There also exists a master program, appropriately called psi3, which will parse an input file, recognize the overall calculation desired, and run all the necessary modules in the correct order. As we will see later, it is possible to define entirely new calculation types simply by editing a text file.

By default, the psi3 program reads its input from a text file called input.dat, and output from PSI3 is written to output.dat. However, it is also possible to specify different input and output filenames by adding these to the command line, like this:

```
psi3 input-name output-name
```

What does a PSI3 input file look like? First of all, due to the modular nature of the program, the input may be split into different sections, each beginning with a keyword and enclosed in parentheses. For now, we will ignore this flexibility and simply put all of the input into a default area called psi:

```
% This is a sample PSI3 input file.
% Anything after a percent sign is treated as a comment.
```

```
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
  )
)
```

As you can see, the input is made up of `keyword = value` pairs. Values can be strings (like `"cc-pVDZ"`), booleans (`true/false, 1/0, yes/no`), integers, or real numbers. One can also have arrays of data (as in the the `zmat` entry). Generally speaking, input is not case-sensitive. Anything following a percent sign is treated as a comment, and white space of more than a single space is ignored.

The above input is for a Hartree-Fock SCF (`wfn = scf`) calculation of water using a cc-pVDZ basis set (`basis = "cc-pVDZ"`). This is a single-point calculation (`jobtype = sp`) at the experimental geometry. Since the ground state of water is closed shell, restricted Hatree-Fock (RHF) orbitals are used (`reference = rhf`). The geometry is specified in the `zmat` array using the standard quantum chemistry Z-matrix input format (for beginners, a short tutorial on Z-matrix inputs is available at `www-rcf.usc.edu/~krylov/AbInitioCourse/zmat.html`). Geometries may also be input in Cartesian coordinates, as we will see later.

Why are some strings in quotation marks (e.g., `label = "cc-pVDZ SCF H2O"`), while others are not (e.g., `wfn = scf`)? Any string containing spaces or other special characters like asterisks must be placed in double quotes. Anything between double quotes is considered one token; there is no change of case, and special characters and white space are maintained as part of the token but otherwise ignored. Since basis sets often contain special characters, it is a good idea to specify the basis in quotes always.

Let's run this calculation. Assuming the input is given in a file `sp.in`, we can run `PSI3` as:

```
psi3 sp.in sp.out
```

This will run the driver program `psi3` which will read the input from `sp.in` and write the output to `sp.out`. The driver program will print something like this:

```
 The PSI3 Execution Driver

PSI3 will perform a RHF SCF energy computation.
```

```
The following programs will be executed:

 input
 cints
 cscf

input
cints
cscf
```

The `psi3` program figures out what type of calculation it is (a restricted Hartree-Fock self-consistent-field energy computation) and it runs the appropriate `PSI3` modules, which are `input`, `cints`, and `cscf`. The `input` program reads the geometry and basis set information from the input file and places this information in a "checkpoint file" (with a filename ending in `.32`) which is used during the calculation. The `cints` module computes the one- and two-electron integrals, and the `cscf` module computes the SCF energy. Since this is all we requested, the program stops here.

What does the output of the program look like? There is quite a lot in the output, so we will focus only on the most pertinent portions. The first part should look something like this:

```
*****************************************************************************
tstart called on aurelius.chemistry.gatech.edu
Tue Aug 26 17:35:10 2003

                                   --------------
                                     WELCOME TO
                                       PSI  3
                                   --------------

  LABEL       = cc-pVDZ SCF H2O
  SHOWNORM    = 0
  PUREAM      = 1
  PRINT_LVL   = 1

  -Geometry before Center-of-Mass shift (a.u.):
       Center              X                  Y                   Z
    ------------   -----------------  -----------------  -----------------
          OXYGEN      0.000000000000     0.000000000000     0.000000000000
        HYDROGEN      0.000000000000     0.000000000000     1.808467771070
        HYDROGEN      1.750863805261     0.000000000000    -0.452804167853

...
```

This first section corresponds to the output from the `input` program, which translates

the basis set and geometry information from the user's input file. At the beginning of each module, timing information is printed which gives the time and date the module is started and the name of the computer running the computation. After parsing the geometry, the program shifts the molecule so that the origin is at the center of mass, and it may also rotate the molecule so that any spatial symmetry in the molecule may be easier to detect. The program then prints out the detected point group of the molecule, which in this case is C2v:

```
-SYMMETRY INFORMATION:
  Computational point group is C2v
  Number of irr. rep.      = 4
  Number of atoms          = 3
  Number of unique atoms   = 2
```

You should double-check that the program correctly identifies the symmetry of your molecule; the PSI3 program is not tolerant of any rounding errors in the user input of geometry.

Next, the basis set information is printed out explicitly in a format identical to that used for customized input of basis sets, as described in section 3.7.2. After this, the program prints a summary of the basis set information, including the number of basis functions in atomic orbitals (AO's) and symmetry-adapted orbitals (SO's), and the number of SO's per irredicible representation (irrep) of the point group:

```
-BASIS SET INFORMATION:
  Total number of shells = 12
  Number of primitives   = 31
  Number of AO           = 25
  Number of SO           = 24

  Irrep    Number of SO
  -----    ------------
    1           11
    2            2
    3            4
    4            7
```

The number of SO's can be very useful to check to verify that the correct type of Gaussian basis functions are used; for example, using 6 Cartesian d functions or 5 spherical harmonics d functions will give the same number of AO's, but a different number of SO's. In this case, we are using Dunning and co-workers' correlation consistent cc-pVDZ basis set, which always utilize spherical harmonics Gaussians, hence the number of SO's is smaller than the number of AO's.

After some additional geometry information, the input program prints the nuclear repulsion energy and then a list of internuclear distances:

```
  Nuclear Repulsion Energy (a.u.) =       9.196934380445
```

```
  -The Interatomic Distances in angstroms:

           1           2           3

    1   0.0000000
    2   0.9570000   0.0000000
    3   0.9570000   1.5133798   0.0000000
```

The nuclear repulsion energy is another very useful diagnostic; when comparing the results of two different calculations, if the calculations are really performed at the same geometry, the nuclear repulsion energies should match.

Once the `input` program completes, a file ending in `.32` should be created. This is the "checkpoint file" which is used to communicate basic information (the basis set, the geometry, etc) between modules in `PSI3`. By default, this file will be placed in the current working directory when the program is run. The name and location of the file can be modified by the user as described later in sections 3.5 and 3.6.

Now that `PSI3` knows about the molecule and basis set, the next module to execute is the `cints` module.

```
                    ---------------------------------------------
                      CINTS: An integrals program written in C
                       Justin T. Fermann and Edward F. Valeev
                    ---------------------------------------------


  -OPTIONS:
    Print level                 = 1
    Integral tolerance          = 1e-15
    Max. memory to use          = 2500000 double words
    Number of threads           = 1
    LIBINT's real type length   = 64 bit
...
```

This program computes integrals over symmetry orbitals. By default, these integrals are written to disk unless the user has specified an integrals-direct computation. The integrals file can be very large, and its size grows rapidly with the size of the molecule and basis set. When `cints` is finished, it prints the number of two-electron integrals:

```
    Wrote 11412 two-electron integrals to IWL file 33
```

Once the integrals have been written, it is time to perform the Hartree-Fock SCF computation using the `cscf` module.

```
          ----------------------------------------

             CSCF3.0: An SCF program written in C

           Written by too many people to mention here

          ----------------------------------------

  I think the multiplicity is 1.
  If this is wrong, please specify the MULTP keyword

  label       = cc-pVDZ SCF H2O
  wfn          = SCF
  reference    = RHF
  multiplicity = 1
  charge       = 0
  direct       = false
  dertype      = NONE
  convergence  = 7
  maxiter      = 40
  guess        = AUTO

  nuclear repulsion energy        9.1969343804448
  first run, so defaulting to core-hamiltonian guess

...
```

The first few lines print out information about the spin multiplicity specified by the user, the convergence level, maximum number of iterations, etc. After this, the program attempts to figure out the symmetries of the occupied orbitals using a core Hamiltonian guess. The `cscf` program sometimes guesses occupations wrong, especially for highly strained molecules, radicals, unusual bonding situations, or large basis sets (particularly with diffuse functions). Unfortunately, this is a common problem with programs which have point-group symmetry built in throughout the program; however, the user can override the guess by specifying these occupations explicitly using the `docc` and `socc` keywords. In this case, the program determines these occupations correctly, as below.

```
  Using core guess to determine occupations


  Symmetry block:   A1    A2    B1    B2
  DOCC:              3     0     1     1
  SOCC:              0     0     0     0
```

The `DOCC` array lists number of doubly-occupied orbitals per irrep, and the `SOCC` array lists

the number of singly-occupied orbitals per irrep (unnecessary for closed-shell molecules). These occupations could have been specified explicitly in user input as:

```
docc = (3 0 1 1)
socc = (0 0 0 0)
```

Once the occupations are available, the self-consistent-field procedure begins to solve the Hatree-Fock equations.

iter	total energy	delta E	delta P	diiser
1	-68.8728484552	7.806978e+01	0.000000e+00	0.000000e+00
2	-71.2986301202	2.425782e+00	2.008202e-01	9.307684e-01
3	-73.8237872940	2.525157e+00	1.056816e-01	9.960987e-01
4	-75.8742610925	2.050474e+00	5.205943e-02	7.105479e-01
5	-76.0217491034	1.474880e-01	1.313776e-02	1.898892e-01
6	-76.0263470429	4.597939e-03	1.920732e-03	5.155862e-02
7	-76.0267968846	4.498417e-04	8.594423e-04	1.595832e-02
8	-76.0268077609	1.087637e-05	1.063540e-04	1.111887e-03
9	-76.0268081461	3.851364e-07	1.914348e-05	2.419943e-04
10	-76.0268081747	2.867948e-08	4.819477e-06	6.654983e-05
11	-76.0268081769	2.160775e-09	1.655548e-06	1.348540e-05
12	-76.0268081769	1.818989e-11	1.534696e-07	1.509549e-06
13	-76.0268081769	1.264766e-12	4.735582e-08	4.720895e-07

For each iteration, the program lists the total electronic energy, the change in energy from the previous iteration, the RMS change in the density matrix, and the maximum element in the DIIS error array. Once the delta P has dropped below 10^{-n}, where n is the number specified as the convergence criterion (by default, 7), the self-consistent-field procedure is considered complete.

At the end of the computation, the orbital energies are reported (in Hartrees), along with their symmetry labels:

```
Orbital energies (a.u.):

  Doubly occupied orbitals
   1A1    -20.550390     2A1     -1.336826     1B2     -0.699406
   3A1     -0.566646     1B1     -0.493169


  Unoccupied orbitals
   4A1      0.185609     2B2      0.256291     3B2      0.789452
   5A1      0.854627     6A1      1.163500     2B1      1.200384
   4B2      1.253301     7A1      1.444438     1A2      1.476370
   3B1      1.674656     8A1      1.867219     5B2      1.935269
   6B2      2.453470     9A1      2.491061     4B1      3.285961
```

```
2A2      3.339124    10A1      3.510949    11A1      3.866084
7B2      4.147878
```

For typical closed-shell molecules, the occupied orbitals should have negative energies, and the unoccupied orbitals should have positive energies, as in this case. The highest occupied orbital (HOMO) is a $1B_1$ orbital with an energy of -0.493169 Hartree, and the lowest unoccupied orbital (LUMO) is the $4A_1$ orbital with energy 0.185609 Hartree.

Frequently, more specialized input will be required. For example, maybe you wish to compute the energy of a radical anion using a large, diffuse basis set. It may be difficult to converge the Hatree-Fock wavefuction for a case like this, and it might require a different initial guess, more iterations, etc. Special keywords for accomplishing such tasks are described later in this user's manual. It is also possible to find this information through the use of the `PSI3` manual pages and the `man` command. For the man pages to be accessible, the user must add them to their `MANPATH` environmental variable (as described in the installation manual). For C shell, this can be accomplished using

```
setenv MANPATH $psipath/doc/man:$MANPATH
```

where `psipath` is the path to your installed version of `PSI3`(e.g., `/usr/local/psi3`). The manual for the `cscf` module can then be accessed simply by typing

```
man cscf
```

which will display a summary of the module and a list of user options.

3 PSI3 Input Files

3.1 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.

3.2 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 `psi3` 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 3.3 and 3.7. 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, DETCAS, CCSD, CCSD_T, BCCD, BCCD_T, EOM_CCSD.

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!

3.3 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`;

3.4 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 porperties 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_{2h} has 3 non-equivalent C_{2v} subgroups, e.g. $C_{2v}(X)$ 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 $C_{2v}(X)$ subgroup of D_{2h} 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}

3.5 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 = "./" )
  )
)
```

3.6 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 = "./")
  )
)
```

3.7 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*.

3.7.1 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.

3.7.2 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:

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

- 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
  ....
)
```

  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.

3.7.3 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 Theoretical Methods Available in PSI3

Several electronic structure methods are available in the `PSI3` package, from Hatree-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.

4.1 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.

4.2 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. Both MP2 and MP2-R12 energies can be computed using integral-direct techniques using multithreaded algorithm, which lends itself perfectly for execution on symmetric multiprocessor (SMP) machines. Table 7 summarizes these capabilities. This section describes how to carry out MP2 and MP2-R12 calculations within `PSI3`.

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

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

4.2.1 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 is `rhf`. 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.

4.2.2 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.

4.2.3 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/).

4.3 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.

Reference	Method	Energy	Gradient	Exc. Energies	LR Props
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	N	–	–
Brueckner	CCD	Y	N	N	N
Brueckner	CCD(T)	Y	N	–	–

Table 8: Current coupled cluster capabilities of `PSI3`.

4.3.1 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.

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

JOBTYPE = string
Acceptable values are `sp`, `opt`, `freq`, or `response` (for coupled cluster linear response calculations). 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`. New algorithms are in progress for significantly speeding up this portion of the computation.

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 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`.

4.3.2 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.

4. Set the `AO_BASIS` keyword to `DISK`. This will save significantly on disk space.

4.3.3 Excited State (EOM-CCSD) 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.

4.3.4 Linear Response (CCLR) Calculations

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

PROPERTY = string
For linear-response calculations, this specifies the type or response property desired. Acceptable values are `POLARIZABILITY` (default) for dipole-polarizabilities and `ROTATION` for specific rotations. For `PROPERTY=ROTATION`, both types of property are actually computed.

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.

4.4 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. It can solve for restricted active space configuration interaction (RAS CI) wavefunctions as described by Olsen, Roos, Jorgensen, and Aa. Jensen. 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.

4.4.1 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
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
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
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
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.

RAS 3 = integer array
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).

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.

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

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

CASSCF is a general method for obtaining a qualitatively correct wavefunction for highly strained molecules, diradicals, or bond breaking reactions. 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 and a Newton-Raphson update, accelerated by Pulay's DIIS procedure. Future versions of the program will allow RASSCF type wavefunctions.

At present, CASSCF's work most efficiently if the inactive orbitals are treated as "frozen" in the `FROZEN_DOCC` and `FROZEN_UOCC` arrays. These orbitals are still optimized by the CASSCF. This procedure is being simplified for the next release of `PSI3`. See the `casscf-sp` example in the `tests` directory for an example of the current input.

4.5.1 Basic Keywords

WFN = string
This should be `detcas` for determinant-based CASSCF.

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
Energies (`none`) and gradients (`first`) are supported.

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.

FROZEN_DOCC = integer array
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

The number of highest energy unoccupied orbitals in each irreducible representation into which there will be no excitations. 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.

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 H2O CISD).

5 Geometry Optimization and Vibrational Frequency Analysis

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 analytic gradients are available (see Table 2), PSI3 will automatically generate and use so-called redundant internal coordinates for carrying out the optimization. These simple stretch, bend, torsion, and linear bend coordinates may be read from the intco.dat file or determined completely by distance criteria using the input geometry, and written by to the intco.dat file. By default, optimization is performed in redundant internal coordinates regardless of whether cartesian or z-matrix coordinates were given in the input. However, optimizations cannot be perfomed if any dummy atoms are used in the specification of the input geometry.

For methods for which only energies are available, PSI3 will use 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 has not been extensively tested. For both analytic- and finite-difference-gradient optimizaiton methods, Hessian updates are performed using the BFGS method.

In addition, simple internal coordinates may be generated from the z-matrix given in the input. The user may denote (using a dollar sign) which coordinates should not be optimized. However, this release of PSI3 does not have a general mechanism to rigorously enforce geometrical constraints, so the results should be examined carefully.

PSI3 is also capable of computing harmonic vibrational frequencies for a number of different methods using either analytic energy first or second derivatives. (At present, only RHF-SCF analytic second derivatives are available.) For those methods for which analytic gradients have been coded (see Table 2), PSI3 will generate displaced geometries along

symmetrized, delocalized internal coordinates, compute the appropriate first derivatives, and use finite-difference methods to compute the Hessian.

For finite-difference procedures for vibrational frequency calculations, the user should keep in mind that, for geometric displacements along non-totally-symmetric coordinates, by definition, the molecular point group symmetry will decrease. As a result, any PSI3 keywords in the input file which depend on the number of irreducible representations will be incorrect. For example, if the user must specify the DOCC keyword in order to obtain the correct MO occupations for their system, computation of non-symmetric vibrational modes via finite-differences will necessarily fail. The developers are working to further automate the computation of vibrational frequencies such that the correlation of irreducible representations between point groups can be handled correctly. This feature will be available in a future release of the package.

The following keywords are pertinent for geometry optimizations and vibrational frequency analyses:

JOBTYPE = string
This keyword (described earlier in this manual) must be set to OPT for geometry optimizations and FREQ for frequency analyses.

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

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.

6 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

6.1 Basic Keywords

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

JOBTYPE = string
This keyword must be set to oeprop for PSI3 to compute electron properties. There is no default.

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.

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.

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`.

6.2 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`.

6.3 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.

6.4 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.

7 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.

7.1 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.

7.2 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.

7.3 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)
)
```

7.4 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.

8 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 `PSI3` 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, PSI 3.2, 2003.