

DIRAC04: documentation

Release DIRAC 04.0 (September 2004)

Program
• for
Atomic
• and
Molecular

Direct
Iterative
Relativistic
All-electron
Calculations



<http://dirac.chem.sdu.dk>
e-mail: dirac-admin@dirac.chem.sdu.dk

Contents

Preface	7
1 DIRAC04 Installation Guide	9
1.1 Hardware and software supported	9
1.2 Installing the program	9
1.3 Maintenance	10
1.3.1 New versions	10
1.3.2 Reporting bugs	11
1.3.3 User support	11
2 User's Guide	13
2.1 Input files	13
2.2 The <code>pam</code> shell script	14
2.3 The <code>pamq</code> shell script	15
2.4 A simple job example	15
2.5 Writing your own shell scripts	16
2.5.1 Overview of environment variables	17
2.6 Test examples	18
3 DIRAC Menu File	19
3.1 **DIRAC — Job specification	21
3.1.1 *OPTIMIZE — geometry optimization directives	22
3.2 **GENERAL — General input	23
3.2.1 Sample input	24
3.3 **INTEGRALS — integral directives	25
3.3.1 *READIN — reading basis set file	26
3.3.2 *ONEINT — one-electron integrals	26
3.3.3 *TWOINT — two-electron integrals	26
3.3.4 Sample input	28

3.4	**HAMILTONIAN — specification of Hamiltonian	29
3.4.1	*DFT — DFT directives	32
3.4.2	Sample inputs	33
3.5	**WAVE FUNCTIONS — get wave function	34
3.5.1	*DHFCAL — Hartree-Fock/Kohn-Sham calculation.	35
3.5.2	Sample input	45
3.5.3	*MP2CAL — MP2 calculation.	45
3.5.4	*RESOLVE — Resolve open-shell states.	47
3.5.5	*MVOCAL — Modified virtual orbitals.	48
3.6	**ANALYZE — analyze wave function	49
3.6.1	*PRIVVEC — Vector print	49
3.6.2	*MULPOP — Mulliken population analysis	50
3.6.3	*PROJECTION — Projection analysis	51
3.6.4	*DENSIT — generate density for visualization	52
3.6.5	*RH01 — Print density along bonds	55
3.6.6	Sample inputs	56
3.7	**PROPERTIES — property module	58
3.7.1	*EXPECTATION VALUE — Evaluate expectation values	60
3.7.2	*LINEAR RESPONSE — Linear response	60
3.7.3	*NMR — Controls evaluation of NMR parameters	63
3.7.4	*MOLGRD — Controls evaluation of the molecular gradient	64
3.7.5	*QUADRATIC RESPONSE — Quadratic Response Functions	64
3.7.6	Input samples	66
3.8	**MOLTRA — integral transformation module	68
3.8.1	*PRPTRA — Property integrals transformation	70
3.8.2	Sample input	70
3.9	RELCCSD — coupled cluster module	71
3.9.1	RELCCSD — Generic and control input	71
3.9.2	CCENER — Input for energy module	72
3.9.3	CCSORT — Input for sorting module	73
3.9.4	CCFOPR — Input for first order properties	73
3.9.5	CCFSPC — Input for Fock space module	74
3.9.6	Sample input for RELCCSD	75
3.10	DIRRCI — direct CI module	76
3.10.1	RASORB — Specify the type of CI and the active spaces	76
3.10.2	CIROOT — Select the state to converge on	76
3.10.3	DIRECT — Convergence control	77
3.10.4	OPTIM — Fine tuning of algorithm	77
3.10.5	LEADDET — Analyze CI wave function	77
3.10.6	Sample input for DIRRCI	78

3.11	GOSCIIP — COSCI module	79
3.11.1	GOSCIIP — Specify the CI space	79
3.11.2	POPANA – analysis of wave function	79
3.11.3	Sample input for GOSCIIP	79
3.12	LUCITA — direct GAS CI module	80
3.12.1	General Input	80
3.12.2	Specific Input	82
3.12.3	Sample inputs for LUCITA	83
3.13	Special features	85
3.13.1	Orbital strings	85
4	DIRAC Basis File	87
4.1	Title and symmetry	87
4.2	Atomic coordinates	89
4.3	Large component basis set	90
4.3.1	Basis sets from the basis set library	90
4.3.2	Explicitly typed basis set	90
4.3.3	MOLFDIR-type basis sets	93
4.3.4	Even-tempered basis sets (geometric progressions)	93
4.3.5	Well-tempered basis sets	94
4.3.6	Family basis sets	95
4.3.7	Dual family basis sets	96
4.4	Small component basis set	98
4.4.1	By kinetic balance	98
4.4.2	Explicitly typed basis set	99
4.4.3	MOLFDIR-type basis sets	99
5	Output Files	101
5.1	Restart	102
6	Specification of One-Electron Operators	103
6.1	Syntax for specification of operators	103
6.2	List of one-electron operators	105
A	Recent Modifications	111

Preface

This is the documentation for the program “DIRAC 04, a relativistic ab initio electronic structure program”.

The source code and the program is provided under a written licence and may be used, copied, transmitted, or stored only in accord with that written licence (see <http://dirac.chem.sdu.dk>). In particular, no part of the source code or compiled modules may be distributed outside the research group of the licence holder. This means also that persons (e.g. post-docs) leaving the research group of the licence holder may not take any part of DIRAC 04, including modified files, with him/her, unless that person has obtained his/her own licence.

For questions concerning DIRAC , this copyright, or information on how to get a licence, see <http://dirac.chem.sdu.dk> or write to dirac-admin@dirac.chem.sdu.dk.

Any use of the program that results in published material should cite the following:

“DIRAC 04, a relativistic ab initio electronic structure program, Release DIRAC04.0 (2004), written by H. J. Aa. Jensen, T. Saue, and L. Visscher with contributions from V. Bakken, E. Eliav, T. Enevoldsen, T. Fleig, O. Fossgaard, T. Helgaker, J. Laerdahl, C. V. Larsen, P. Norman, J. Olsen, M. Pernpointner, J. K. Pedersen, K. Ruud, P. Salek, J. N. P. van Stralen, J. Thyssen, O. Visser, and T. Winther.”

The program represents experimental code and is under constant development. This means that there is no warranty of correctness of results nor fitness for a particular purpose.

Acknowledgments

We would like to acknowledge the basis set development and testing by K. G. Dyall, K. Faegri Jr., and A. S. P. Gomes. We also acknowledge porting and beta-testing by S. Dubillard, U. Ekström, I. Infante, J. Henriksson, and C. R. Jacob.

Chapter 1

DIRAC04 Installation Guide

1.1 Hardware and software supported

DIRAC can be run on a number of different UNIX operating systems. The DIRAC04 release has been tested on Linux, SGI, IBM AIX, Apple OsX, Cray, HP, Sun, Dec, and others.

Most of the program is written in Fortran 77, although the main program and some utility routines are written in C. Machine-dependent (compiler-dependent) features are handled with the C-preprocessor. All floating point computations are performed in double precision (64-bit).

DIRAC04 may work in other UNIX environments, and the user is encouraged to report back any successful porting of the program.

DIRAC04 has not been ported to any WINDOWS operating system.

1.2 Installing the program

The program can be installed on a number of different types of computers. It is present as a series of *.F master files that are processed by the cpp C preprocessor to handle machine-specific features. The installation is handled by the UNIX make command generator.

The program is installed as follows:

- The program system is supplied as a tar-file compressed with gzip: DIRAC04.tar.gz. Unpack the program system by the commands

```
tar -zxvf DIRAC04.tar.gz
```

or, on some machines

```
gunzip DIRAC04.tar.gz  
tar xvf DIRAC04.tar
```

and a directory DIRAC04 has been created with the source code.

- `cd DIRAC04`
- Check content of the README file.
- Now run the `configure` script
 - `./configure`
 - The script should automatically guess your machine/architecture
 - `configure` makes the file `Makefile.config` with system dependent information (as which compilers, libraries, compiler flags etc. to use) for `make` and the `pam` shell script for starting DIRAC calculations.
 - For help on `configure`:
`configure --help`
Note that `configure` can be invoked with flags, e.g. for choosing compilers or paths for mathematical libraires.
- Check `Makefile.config` and edit if needed.
- If you want to use a queuing system: check `pamq` and edit if needed.
- Optional: Run `make depend`:
`make depend`
This finds out dependencies for each file, that is, if you change a header file, then all Fortran and C files using this header file will be recompiled. This is very useful for programmers, or if you wish to change some dimensions in a common block.
- Run `make`:
`make`
for compilation/installation.

1.3 Maintenance

1.3.1 New versions

New versions will be released if bugs are found and repaired and when new features become available. These updates will be announced on the mailing list and they will be available for download on DIRAC homepage.

DIRAC is under constant development and some new features might not be in the release that you have access to. If you have interest in unreleased features developed in the DIRAC author group, you are asked to contact the author(s) directly. If the authors

decide to hand over some new code, this may be under restrictions not mentioned in the licence agreement.

1.3.2 Reporting bugs

Users are encouraged to report bugs etc. either to author of the “bugged” module or to `dirac-admin@dirac.chem.sdu.dk`.

1.3.3 User support

No user support is organized — although a mailing list is available at (`dirac@dirac.chem.sdu.dk`). All licence holders are added to the mailing list. Additional subscription: mail to `dirac-request@dirac.chem.sdu.dk` with subject: “Subscribe”. The authors also receive the mail from the mailing list, but we do not guarantee that any author will reply to questions posted to the mailing-list. Rather, we encourage the users to help out each other. Please check the archive of previous messages, which is available under <http://dirac.chem.sdu.dk/>.

Chapter 2

User's Guide

2.1 Input files

Two input files are needed to run DIRAC :

- The **basis file** defines the basis set, nuclear configuration and symmetry.

- The **menu file** defines the calculation.

At the start of any calculation the basis file is processed and then various modules are activated based on the information given in the menu file. One may therefore run a sequence of calculations based on the same basis file. In such cases it is then recommended to keep the file with MO-coefficients, DFCOEF, that makes it possible to restart the calculation. For instance, one may first run a DC-HF-calculation which gives a set of MO-coefficients defining the converged wave function. Population analysis may be performed in a separate calculation and then requires only the file of coefficients in addition to the basis file.

2.2 The pam shell script

The script `pam` is available for automatization of calculations. Invoke `pam` without any flags to get a list of flags, the output should look like the following (with local modifications):

```
Usage: pam [Flags] mol[.mol] menu[.inp]
Flags:
  -mpi n                : Run MPI with n nodes; default sequential
  -machinefile filename : Specify file with list of machines to be used in MPI run
  -mpiarg string        : Any other arguments to be passed on to MPI run
  -mw mem               : Set memory (in megawords) for master
                        (and slave nodes if "-nw mem" not specified)
  -nw mem               : Set memory (in megawords) for the slave nodes
  -basdir directory    : Prepend basis set directory to search list (may be repeated)
  -wrkdir directory    : Directory for work area; default "/scr/jth/.$$"
  -clean               : Clean work area before calculation
  -keepwrk             : Keep work area after calculation (default: remove)
  -copy "file1 file2 .." : Copy specified files from job directory to work area
  -get "file1 file2 .."  : Get specified files to job directory from work area
  -incmo               : Copy DFCOEF from job directory to work area (when restarting)
  -outcmo              : Save DFCOEF to job directory from work area (for restart)
  -infck               : Copy DFFCK2 from job directory to work area
  -outfck              : Save DFFCK2 to job directory from work area
  -dirac dirac_cmd     : Use dirac_cmd as executable instead of the default dirac.x
  -run name             : File suffix for output; default "out"
  -nice                : execute Dirac with nice/npri
  -notify              : Notify when finished to user starting job
  -user e-mail         : Notify when finished using specified e-mail address
  -debug               : Run dirac in debugger "gdb"
  -ssrun exp_type      : use ssrun exp_type (SGI profiler) to execute Dirac
                        (path must be included if executable is not in
                        input directory)
  -help                : More help on pam than this
```

Using the `pam` script a job with the input files `Au2.mol` and `ccsd.inp` may be started with the following shell script

```
pam Au2 ccsd
```

An example of a restart is the following DFT calculation on iodobenzene where the MO-coefficients are kept for analysis, using the input files `C6H5I.mol`, `B3LYP.inp` and `analysis.inp` :

```
pam -outcmo C6H5I B3LYP
pam -incmo C6H5I analysis
```

2.3 The pamq shell script

One often will want to submit the `pam` script to a batch queuing system rather than running `dirac` interactively as is done with `pam`. To simplify running `DIRAC` in such an environment one may use the `pamq` script that will generate the proper job file and take care of the submission of the job. This script takes the same arguments as `pam` but has a few additional arguments relating to the maximum amount of time that should be used and the specific queue that the user may want to select. Invoke `pamq` without any flags to get a list of flags, the output should look like the following but with your local defaults:

```
Usage: pamq [pamflags] [pamqflags] mol[.mol] menu[.inp]
  pamflags :          Invoke pam without options to get a list of flags for pam
  pamqflags:
    -q queue          : Submit job to the specified queue; default is "default"
    -qsys queuesys   : Which queue system to use; default is "pbs"
                      Known queue systems: pbs, bsub, LoadLeveler, XGrid
    -t time           : Time limit for job; default is "15m"
                      Specify either hours (as in 20h) or minutes (as in 30m)
    -mpi n            : Run MPI with n nodes; default is 8
    -mw mem           : Set memory (in megawords) for master; default is 100
                      (and slave nodes if -nw mem not specified)
    -nw mem           : Set memory (in megawords) for the slave nodes; default is 100
```

Running

```
pamq -mpi 6 -t 2h C2H4Cl2_ec1_c2v lda
```

will submit a job script to the default queue with a time limit of 2 hours and requesting 6 processors. The input files could for example be the files described in the next section (although this example doesn't really need 6 nodes!).

2.4 A simple job example

This section will illustrate how to specify a DFT calculation on dichloroethane in an eclipsed conformation. In this example, the `DIRAC` basis file `C2H4Cl2_ec1_c2v.mol` is (lines beginning with `#` are comments)

```
DIRAC
1,2-dichloro-ethane in eclipsed_1 conformation (Cl eclipsed), angles sp3
Full symmetry : C2v
# Number of different atoms, symmetry specification
C 3 2X Y A
```

```
# Atom 1: charge and number of distinct atoms of this type
      6.    1
# Name of atoms and coordinates
C   -0.7000 0.0000 0.0000
# Basis set (taken from the library)
LARGE BASIS cc-pVDZ
      17.    1
Cl  -1.3000 0.0000 1.6971
LARGE BASIS cc-pVDZ
      1.     1
H   -1.0667 0.8981 -0.5185
LARGE BASIS cc-pVDZ
FINISH
```

(This basis file is also used in test 29 in the directory `test`.) The precise format for the basis file is given in chapter 4. The DIRAC input specification in menu file `lda.inp` is

```
**DIRAC
.TITLE
LDA energy calculation
.WAVE F
**HAMILTONIAN
# Use Simple Coulombic Correction to model effect of (SS|SS) integrals
# This is highly recommended for the calculation of spectroscopic
# constants and valence properties
.LVCORR
# Use Density Functional Theory
.DFT
LDA
**WAVE FUNCTIONS
.DHF
*END OF INPUT
```

2.5 Writing your own shell scripts

Some users will prefer to write their own shell scripts to run DIRAC04 . The details for running the executable `dirac.x` directly are given below. The basic UNIX command for running the program is

```
dirac.x > {outputfile}
```

The menu file must be present as `DIRAC.INP` and the basis file must be present as `MOLECULE.INP`.

The various program modules have different memory requirements. The available memory may be modified at runtime by setting an environmental variable `DIRWRK` which defines the number of 8-byte words available for the calculation. For example,

```
setenv DIRWRK 30000000
```

means that 30 Mw of memory are to be allocated¹. For parallel jobs the memory for the nodes can be set with the similar variable `DIRNOD`. Note that it suffices that these environmental variables are given to the master; it will pass them on to the slaves. The parallel version can run either parallel or sequential jobs. If the variable `DIRPAR` is explicitly set to zero, then `dirac.x` is executed sequential, otherwise parallel. Default memory is specified in the `configure` script.

2.5.1 Overview of environment variables

DIRWRK Size of memory in 8-byte words for master node, and for slaves if no `DIRNOD` is given. For example,

```
setenv DIRWRK 30000000
```

to allocate 30 MW, or approx. 228 MB of memory.

DIRNOD Size of memory in 8-byte words for slave nodes. Since the memory requirements for the master node is typically higher than for the slaves, it is often useful to allocate less memory for the slaves than for the master. This is useful for cluster-type computers such as IBM SP or Beowulf where you might run the master on a “fat” node with lots of memory and the slaves run on ordinary nodes, but it is also useful for shared memory computers such as SGI Origin to reduce the total memory allocation. In CC runs one should NOT choose `DIRNOD` different from `DIRWRK` because the parallelization assumes identical nodes and does not use a master-slave algorithm. The code will work but performance will be suboptimal due to load balance problems.

DIRPAR A parallel version of `DIRAC` can be run in serial mode (without calling any MPI routines) if `DIRPAR` is set to zero, i.e.

```
setenv DIRPAR 0
```

It will run in parallel if `DIRPAR` is not set or is set to a non-zero value.

¹1 Mw = 8,000,000 bytes = 7.63 MB. 1MB = (1024)² bytes.

BASDIR The path where DIRAC searches for basis sets. BASDIR may contain many directories separated by colons. The default path in the pam shell script for a user jth is (when DIRAC04 is installed in /home/jth/prog/DIRAC04) :

```
setenv BASDIR " ./home/jth/prog/DIRAC04/basis:/home/jth/prog/DIRAC04/basis_dalton"
```

A shell script for running DIRAC by user jth on the job example described in section 2.4 might look like

```
#!/bin/tcsh

setenv SCR /scr/jth

cp lda.inp $SCR/DIRAC.INP
cp C2H4C12_ec1_c2v.mol $SCR/MOLECULE.INP

setenv DIRWRK 1000000

cd $SCR

$HOME/DIRAC04/dirac.x > exciting_output
```

2.6 Test examples

DIRAC is provided with a number of test cases. These test jobs are placed under `./test` in the DIRAC04/ directory. In the DIRAC04/test/ directory a number of subdirectories exists, each containing a test job and a README file, with a description of the test. These test jobs can also provide you with a starting point for doing calculations yourself.

All tests can be run automatically using the `testlast` script which provides a check on correct installation. This script can be invoked by simply typing

```
testlast
```

in the main directory. It will then run interactively and create logs of all runs in the directory `test/logs`. Check the file `test/logs/lastrun` to see if any errors were detected. This run should complete within 5 minutes on all reasonable machines. More advanced tests can be run by specifying options to `testlast`. Type

```
testlast -help
```

for an overview.

Chapter 3

DIRAC Menu File

The DIRAC menu file defines the calculation and has the general structure:

```
**DIRAC
  <Required: job specification; specify which modules to activate,
    for example: wave functions, properties, or analysis.>
**GENERAL
  <Optional: general input and description of system under study,
    for example: the speed of light.>
**HAMILTONIAN
  <Optional: specify electronic Hamiltonian, for example:
    spin-free Hamiltonian.>
**INTEGRALS
  <Optional: directives to HERMIT integral generator.>
**WAVE FUNCTION
  <Optional: specification of wavefunctions if module is activated
    under **DIRAC.>
**ANALYZE
  <Optional: specification of wave function analysis if module is
    activated under **DIRAC.>
**PROPERTIES
  <Optional: specification of property calculation if module is
    activated under **DIRAC.>
**MOLTRA
  <Optional: change of default values for the 2- and 4-index integral
    transformations if this module is active.>
**END OF DIRAC
```

The menu file is divided into various sections all beginning with a keyword of the form **XXXXXX**. Only the first section ****DIRAC** is mandatory. Some sections (****HAMILTONIAN**) will be read if present; the others may be invoked by keywords in the ****DIRAC** section. Each section has the general structure:

```
**XXXXX  
.<keywords>  
*<subsection>  
.<keywords>  
*<subsection>  
.<keywords>  
*<subsection>  
.<keywords>  
# a comment  
! another comment  
...
```

Each subsection is referred to by a keyword of the form ***XXXXXX**. For each subsection a set of keywords may be specified, possibly with additional arguments. The set of subsections and keywords allows the user great flexibility in defining the current calculation.

Comment lines (! or # in the first column) are ignored. Only six characters are used of each keyword, but more may be specified for readability.

All sections, subsection, and keywords are documented in the following sections. The keywords are grouped as “Basic options”, which is the required or most used keywords, “Advanced options” for optional keywords, and “Programmers options” which activates debug or experimental features.

Almost every section or subsection has a print level flag which controls how much output a certain module gives. The standard values are set to give sufficient output, and changing these is not recommended as it might produce massive output. Typically a print level of 5 will print very detailed information like complete Fock matrices, etc.

3.1 **DIRAC — Job specification

Basic options As default DIRAC only activates the generation of the basis set (e.g. kinetic balance conditions for small component functions) and the one-electron modules. Two-electron integrals over the atomic basis functions will be calculated when needed by the job modules given below¹.

- .WAVE FUNCTION activates wave function module(s). This activates the reading of the **WAVE FUNCTION section, where the desired wave function type(s) must be specified (default: none).
- .ANALYZE activates the Hartree-Fock analysis module. This activates the reading of the **ANALYZE section.
- .PROPERTIES activates the property module (which will call the integral module for property integrals). This activates the reading of the **PROPERTIES section.
- .OPTIMIZE activates geometry optimization. This activates the reading of the *OPTIMIZE subsection.
- .4INDEX explicitly activates transformation of integrals to molecular orbital basis. This activates the reading of the **MOLTRA section. These transformed integrals are currently only used by the RELCCSD , DIRRCI , and LUCITA modules, and if one of these three modules are requested under **WAVE FUNCTION , then this flag is automatically activated unless .NO4INDEX is specified in this input module. By default molecular orbitals with orbital energy between -10 au and +20 au are included, this can be modified in the **MOLTRA input section.
- .NO4INDEX do not automatically activate integral transformation to molecular orbital basis if any of RELCCSD , DIRRCI , and LUCITA are requested under **WAVE FUNCTION .

Advanced options

- .TITLE title line
Arguments: Title line (max. 50 characters)
Default: DIRAC: No title specified !!!

¹We recommend that you use the DALTON program package, see <http://www.kjemi.uio.no/software/dalton/dalton.html>, if you want to e.g. save the two-electron integrals on disk for another purpose, this is not possible with DIRAC .

- `.INPTEST` input test: no job modules called
Default: Do not activate input test
 It is often useful to start a new set of calculations with an input test in order to check that input file processing is correct before submitting your job.
- `.ONLY INTEGRALS` stop after the calculation of the one-electron integrals for the Hamiltonian and the one-electron integrals specified under `**INTEGRALS`. The integrals are written to disk.

3.1.1 *OPTIMIZE — geometry optimization directives

This section controls the geometry optimization. The geometry optimization algorithm is based on the one of DALTON . The directives are therefore the same as in DALTON (see e.g. <http://www.kjemi.uio.no/software/dalton/dalton.html>), except for these few changes:

- No second-order algorithms are available since the molecular Hessian is not implemented
- a few new keywords (see below)

If `.PROPERTIES` (`.ANALYZE`) in the job input section is specified together with `.OPTIMIZE` then the property (analyze) module is called in each optimization iteration and at the converged geometry. In each iteration the properties (analysis) requested under `**PROPERTIES` (`**ANALYZE`) are calculated, and at the converged geometry the properties (analysis) given in `**PRP F` (`**ANA F`) are calculated.

- `.NO SKIP` See page 64 for the description of `.TRICK` under the `*MOLGRD` section. The "trick" is by default activated in a geometry optimization, since when the current geometry is far away from the equilibrium, e.g. the norm of the gradient is, say, 1.0, then there is no need to calculate the LS and/or SS two-electron gradient because they have a norm of, say 0.001. This keywords forces the LS and SS two-electron gradient to always be evaluated in all geometry iterations (depending on integral flag).
- `.NUMGRA` Force the use of numerical gradient in geometry optimization, e.g. in Hartree-Fock calculations.
- `.TWOGRD` See page 64 for the description under the `*MOLGRD` section.
Default: `TWOGRD(1) = TWOGRD(2) = TWOGRD(3) = 1`

3.2 **GENERAL — General input

This section allows the specification of global characteristics of the current run.

Basic options

Advanced options

- .ACMOUT** dump coefficient in C_1 symmetry to unformatted file `DFACMO`. This can be useful, e.g. when doing a projection analysis: The molecule is calculated at its highest symmetry, each constituent atom type is calculated in linear symmetry (specifying no symmetry operations in the basis file, see chapter 4), and then the analysis is carried out in C_1 symmetry.
- .DIRECT** direct evaluation of two-electron integrals for Fock matrices. (All two-electron integrals for other uses – e.g. CI, CCSD, MCSCF – are always evaluated direct, i.e. never read from disk.)
Arguments: Integers `ILL`, `ISL`, `ISS`
`IXX = 1(on)/0(off)` (`XX = LL,SL or SS`)
Default: `ILL = ISL = ISS = 1`
- .DOJACO** use the Jacobi method for matrix diagonalization
Default is to use the Householder method, which is generally more efficient. The Householder may, however, mix degenerate eigenvectors of different symmetries. The Jacobi method is currently limited to real groups.
Default: Do Householder diagonalization.
- .PCMOIN** Read MO coefficients from formatted file `.DFPCMO`. This is useful for porting coefficients between machines with different binary structure, for example, between Silicon Graphics and IBM. See also keyword `.PCMOUT`.
- .PCMOUT** Write DHF MO coefficients to formatted file `.DFPCMO`. This is useful for porting coefficients between machines with different binary structure, for example, between Silicon Graphics and IBM. See also keyword `.PCMOIN`.
- .LINDEP** thresholds for linear dependence in large and small components; refer to the smallest acceptable values of eigenvalues of the overlap matrix
Arguments: Real `STOL(1)`, `STOL(2)`
Default: Large: `STOL(1) = 1.0D-6`.
Small: `STOL(2) = 1.0D-8`

.SPHTRA transformation to spherical harmonics embedded in transformation to orthonormal basis; totally symmetric contributions deleted.
Arguments: Integers ISPHL,ISPMS = 1(on)/0(off)
Default: ISPHL = 1,ISPMS = 1. ISPHL and ISPMS directs the spherical transformation of large and small components, respectively. The transformation of the large components is a standard transformation from Cartesian functions to spherical harmonics. The transformation of the small components is modified, however, in accordance with the restricted kinetic balance relation.

Programmers options

.PRINT general print level
Arguments: Integer IPRGEN
Default: IPRGEN = 0

.CVALUE set the value of light in atomic units
Arguments: CVAL
Default: CVAL = 137.0359998

3.2.1 Sample input

Below is a sample input of this section:

```
**GENERAL
# Use a reduced value for the speed of light to amplify
# the effect of relativity (for testing)
.CVALUE
50.0
```

3.3 **INTEGRALS — integral directives

This section gives directives to the HERMIT integral code.

Basic options

.NUCMOD nuclear model

Arguments: Integer INUC

INUC = 1: point nucleus

INUC = 2: Gaussian charge distribution.

Default: INUC = 2

The point nucleus model is useful to compare Lévy-Leblond type calculations with regular nonrelativistic calculations done with another code, e.g. DALTON. The two methods should give precisely the same energies.

For the Gaussian charge distribution the default exponents are in accordance with values proposed by Visscher and Dyall in Ref. [1], see also the web-site <http://dirac.chem.sdu.dk/doc/FiniteNuclei/FiniteNuclei.shtml>. If desired, other exponents may be specified in the basis file (the QEXP described on page 89).

Advanced options

.SELECT restrict range of nuclei in one-electron integrals involving single atomic centers, for example electric field gradients.

Arguments: Integer NPATOM — number of atomic centers

Arguments: if NPATOM is non-zero: (IPATOM(I), I=1, NPATOM)

Default: Use all atoms.

.DIPORG origin for all moment integrals, including dipole (dipole is independent of origin only for neutral systems)

Arguments: Real (DIPORG(I), I=1,3)

Default: Origo (0,0,0)

.MAGCOR print symmetrized nuclear magnetic moments; this corresponds to taking symmetry combinations of rotations, not coordinates, at each nuclear center. The numbering is used in labels of various magnetic integrals.

Programmers options

.PRINT general print level in HERMIT

Default: 1

3.3.1 *READIN — reading basis set file

This subsection allows changes of defaults in the reading of the basis file (see Chapter 4).

Advanced options

.UNCONTRACT decontract basis sets specified as contracted (in the library)
 Default: decontract only the small component

Programmers options

.PRINT print level in the reading of the basis file
 Default: print level from **INTEGRALS

.MAXPRI maximum number of primitive functions in a given block in basis file
 Arguments: Integer MAXPRI
 Default: MAXPRI = 22

3.3.2 *ONEINT — one-electron integrals

This subsection gives directives for the generation of one-electron integrals. Based on input in the other section, the program will determine what integrals to calculate.

Programmers options

.PRINT print level in one-electron integral routines
 Default: print level from **INTEGRALS

3.3.3 *TWOINT — two-electron integrals

This subsection gives directives for the generation of two-electron integrals. It also gives directives for the construction of Fock matrices, such as screening.

Advanced options

.SOFOCK do direct Fock matrix construction in symmetry-adapted (SO)- basis.
 Default: direct Fock matrix construction in AO-basis using skeleton matrix approach. This may give better screening, but is more memory intensive.

.ICEDIF separate screening of Coulomb and exchange contributions [2]
 Arguments: Integers ICDIFF(Coulomb),IEDIFF(Exchange) on(1)/off(0)
 Default: ICDIFF = IEDIFF = 1

- .SCREEN** screening threshold for integral direct calculations of Fock matrices [2]
Argument: Real SCRFCK
Default: SCRFCK = 1.0D-12. Note that the screening threshold may influence convergence. In general the screening threshold must be about three orders of magnitude smaller than the desired norm of the electronic gradient at convergence.
- .THRFACT** adjust integral thresholds for SL- and SS-integrals. For conventional integral calculations only integrals above the threshold given in the MOLECULE.INP (see chapter 4 file is written to disk. The thresholds for the SSSL and SSSS integrals are divided by THRFACT's given here.
Arguments: Reals THRFACT(1), THRFACT(2)
Default: THRFACT(1) = THRFACT(2) = 1

Programmers options

- .PRINT** Set print level in two-electron integral routines for the calculation of a particular shell quadruplet. The print level is changed only for the given shell quadruplet. A zero matches all shells, thus

```
.PRINT  
4 0 0 0 0
```

or just

```
.PRINT  
4
```

set print level to 4 for all shell quadruplets. Use with care to avoid massive output !!! At print level 15 the individual integrals are printed.

Arguments: 1-5 integers (IPRINT, IPRNTA, IPRNTB, IPRNTC, IPRNTD)
Default: print level from **INTEGRALS and IPRNTx=0

- .TIME** give detailed timing for integral calculation
Default: No timing.

3.3.4 Sample input

```
**INTEGRALS
#Use point nucleus model to compare with nonrelativistic programs
.NUCMOD
1
```

3.4 **HAMILTONIAN — specification of Hamiltonian

This section defines the electronic Hamiltonian that is to be used. Internally the program will always work with 4-component operators that are expanded using distinct large and small component basis sets. In the transformation to an orthogonal basis set one may, however, combine large and small component functions and/or functions of different symmetry in order to obtain a matrix expansion of e.g. the spinfree modified Dirac equation or the Lévy-Leblond equation [3]. The default Hamiltonian is the unmodified Dirac-Coulomb Hamiltonian. In addition one can also modify the Hamiltonian by introduction of an additional operator, e.g. describing an external field. Any additional operator defined in this section must be totally symmetric under both the molecular point group and time reversal symmetry. The latter requirement precludes the introduction of external magnetic fields.

Basic options

- .LEVY-LEBLOND Use the Lévy-Leblond [4] Hamiltonian. Use this option before any additional one-electron operators are specified, because it redefines the metric used in the calculation.
- .SPINFREE Use Dyalls [5] spinfree Hamiltonian to obtain results without spin-orbit coupling. Note that this option should not be used for response calculations with time-antisymmetric (magnetic) operators as it will eliminate important contributions.
- .LVCORR Model interatomic SS-integral contribution by classical repulsion of small component atomic charges[6]. This is currently the most economical and accurate approximation to the full Dirac-Coulomb Hamiltonian and can certainly be used for the calculation of spectroscopic constants and valence properties; for core properties testing is recommended.
- .DFT Perform a Kohn-Sham calculation and specify a DFT functional.
Predefined functionals: LDA, BLYP, B3LYP
Compose a functional : GGAKEY fun1=w1 fun2=w2 ...
The functional can either be selected from a set of predefined combinations of exchange and correlation functionals, or composed by specifying a list of functionals together with their weights.

List of functionals:

1. Exchange functionals

- LDA exchange [7]: SLATER

- Becke 1988 [8]: BECKE
- The correction term to LDA proposed by Becke [8]: B88CORR
- Perdew-Wang 1986 [9]: PW86C

2. Correlation functionals

- The Vosko-Wilk-Nusair LDA correlation energy (VWN5) [10]: VWN
- The Lee-Yang-Parr functional [11] : LYP
- Perdew 1986 [12]: P86C

Advanced options

- .OPERAT** specification of an additional one-electron operator in the Hamiltonian. The operator must be totally symmetric both under the molecular point group and time reversal symmetry. The field strength of the operator is specified with **COMFACTOR**. The keyword can be repeated for addition of more than one operator. See section 6.1 for more information
- .INTFLG** specify what two-electron integrals to include. All other modules use this **.INTFLG** as the default value.
Arguments: Integers **ILL**, **ISL**, **ISS**
IXX = 1(on)/0(off) (**XX** = **LL**, **SL**, or **SS**)
Default: **ILL** = **ISL** = **ISS** = 1
- .LVNEW** Modification of option **.LVCORR** that obtains the atomic small component charge via a Mulliken analysis instead of the original table look-up. (The problem with the table look-up is that the electrostatics in the molecule will be wrong if you have specified a basis set which does not give the correct small-electron charge because of deficiencies in the core region.)
- .ZORA** Use the Zeroth Order Regular Approach [13, 14, 3] in the Hartree-Fock procedure. The implementation offers only little computational advantages and is intended chiefly for comparisons of methodology. Note that combination **.SPINFREE** and **.ZORA** gives a spinfree formalism that differs from the conventional spinfree ZORA formulation. Two integers should be specified in free format on the line following **.ZORA**. The first indicates whether the density is to be normalized over the 2-component (0 : ZORA) or 4-component metric (1 : ZORA4). The second indicates whether the orbital energies should be unmodified (0 : normal ZORA) or scaled (1 : scaled ZORA).
- .HFXFAC** - weight of exchange in Fock matrix construction
Arguments: Real **HFXFAC**
Default: **HFXFAC** = 1.0D0

Programmers options

- .PRINT** print level
Arguments: Integer IPRHAM
Default: IPRHAM = 0
- .ONESYS** ignore two-particle interactions
Default: ONESYS = .FALSE.
- .FREEPJ** project out all free positronic solutions from the MO-space
Default: .FALSE.
- .URKBAL** unrestricted kinetic balance
Default: Restricted kinetic balance. This is imposed by deleting unphysical solutions from the free particle positronic spectrum. This leads to a 1:1 ratio of electronic and positronic solutions. This preprojection is sensitive to linear dependencies and should therefore preferably be used in conjunction with the spherical transformation of both large and small components.
- .VEXTPJ** project out all external field positronic solutions from the MO-space
Default: No projection
- .NOSMLV** delete SS nuclear attraction integrals. This will take out contributions to the one-electron spin-orbit interaction and Darwin interaction.
Default: Do not delete SS nuclear attraction integrals.
- .SMLV1C** Neglect potential for multi-center SS-blocks, i.e. multi-center small-small nuclear attraction integrals and multi-center SSSS two-electron integrals are neglected.
Default: Do not delete SS nuclear attraction integrals.
- .ONECAP** Consider this taking the SMLV1C model one step further. Only one-center contributions to the LS- and SS-block of two-electron integrals and SS-block of the nuclear attraction integrals are calculated explicitly. The electrostatic effects of the terms neglected this way are included by calculating the classical repulsion from small component charges based on a Mulliken population analysis. Note that we therefore only need to calculate the derivative of the LL-block of integrals when calculating the molecular gradient.
Default: ONECAP = .FALSE.

- `.ONECNV` Employ the one-center model as given by `ONECAP` until convergence to a specified threshold, whereafter the full set of two-electron integrals will be used. This threshold applies to whatever convergence criteria has been selected (`EVCCNV`, `ERGCNV`, `FCKCNV`).
- Arguments:* Real `ONECNV`

3.4.1 *DFT — DFT directives

The DFT module is described in references [15, 16]. It uses standard non-relativistic functionals (see above) since various studies [17, 18, 19] indicate that relativistic corrections to the exchange-correlation functionals have a negligible effect on spectroscopic constants; for core properties further studies are necessary. The numerical integration scheme uses the Becke partitioning [20] of the molecular volume into atomic ones, for which the quadrature is performed in spherical coordinates. Radial integration is carried out using the scheme proposed by Lindh *et al* [21], while the angular integration is handled by a set of highly accurate Lebedev grids. **Note** that the Becke partitioning scheme generally uses Slater-Bragg radii for atomic size adjustments. For the heavier elements, with their more variable oxidation states, this can lead to errors. If the user invoke the `.ATSIZE` keyword, DIRAC tries to deduce relative atomic sizes from available densities, if it can find coefficients.

In the subsection `*DFT` the user can modify parameters for numerical integration. Do not change the default integration parameters unless you know what you are doing.

- `.PRINT` print level
- Arguments:* Integer `IPRDFT`
- Default:* `IPRDFT = 2`
- `.RADINT` Specify maximum error in the radial integration.
- Arguments:* Real `RADERR`
- Default:* `RADERR = 1.0D-13`
- `.ANGINT` Specify the precision of the Lebedev angular grid. The angular integration of spherical harmonics will be exact to L-value `NSCHEME`. By default the grid will be pruned.
- Arguments:* Integer `NSCHEME`
- Default:* `NSCHEME = 41`
- `.NOPRUN` Turn off pruning of the angular grid.
- `.ANGMIN` Specify the minimum precision of the Lebedev angular grid after pruning. Close to the nucleus, the precision of the angular grid will be less than `NSCHEME`, but it will not be less than `LEBMIN`. Note that giving a `LEBMIN`

value greater than or equal to NSHEME is equivalent to turning the pruning off.

Arguments: Integer LEBMIN

Default: LEBMIN = 9

.ATSIZE Generate new estimates for atomic size ratios for use in the Becke partitioning scheme. Relative sizes for a pair of atoms A and B are calculated from their contribution to the large component density along line connecting A and B.

.ERRELS Specify accepted error in the integrated number of electrons. The program stops when larger errors occur. The test is turned off in the first SCF iteration when starting on trial vectors since they will not be orthonormal if they were generated at another geometry.

Arguments: Real DFTELS

Default: DFTELS = 1.0D-3

Advanced options

.INTCHK Test the performance of the grid by computing the overlap matrix numerically and analyzing the errors. In addition, the error matrix can be printed.

Arguments: Integer INTCHK

Output: None (INTCHK = 0), error analysis (INTCHK = 1) and additionally print error matrix (INTCHK = 2)

Default: INTCHK = 0

.FROMVX In Kohn-Sham calculations the exchange-correlation potential is generally not calculated explicitly, only implicitly, that is matrix elements over it, see [15], since this leads to more efficient calculations. **.FROMVX** generates the explicit exchange-correlation potential.

3.4.2 Sample inputs

Add dipole perturbation of strength 0.01 au in the z-direction to the non-relativistic Lévy-Leblond Hamiltonian.

```
**HAMILTONIAN
.LEVY-LEBLOND
.OPERATOR
ZDIPLN
COMFACTOR
0.01
```

3.5 **WAVE FUNCTIONS — get wave function

This section allows the specification of which wave function module(s) to activate. By default no modules are activated. To activate any of these modules you must also specify `.WAVE F` under `**DIRAC`, otherwise this input is not read.

Note that the order below specifies the order in which the different modules are called if you ask for more than one.

Basic options

- `.DHF` activates the Hartree-Fock/Kohn-Sham module. Specification of the DHF module can be given in the `*DHFCAL` subsection. If `.DFT` has been specified under `**HAMILTONIAN` then a Kohn-Sham calculation will be performed, otherwise a Hartree-Fock calculation will be performed.
- `.RESOLVE` Resolve open-shell states: do a small CI calculation to get the individual energies of the states present in an average-of-configurations open-shell Hartree-Fock calculation. See `**RESOLVE` below.
- `.MP2` activates second-order Møller-Plesset module. Specification of the MP2 module can be given in the `*MP2CAL` subsection.
- `.MVO` Calculate modified virtual orbitals. See `*MVOCAL` below.
- `.RELCCSD` activates the coupled cluster module (and the MOLTRA module to get 4-index transformed integrals). Specification of input for the RELCCSD module is given in the *namelist* RELCCSD (see chapter 3.9). By default molecular orbitals with orbital energy between -10 au and +20 au are included, this can be modified in the `**MOLTRA` input section.
- `.DIRRCI` activates the MOLFDIR CI module (and the MOLTRA module to get 4-index transformed integrals). Specification of input for the DIRRCI module is given in the *namelists* DIRRCI and GOSCIP (see chapters 3.10 and 3.11). By default molecular orbitals with orbital energy between -10 au and +20 au are included, this can be modified in the `**MOLTRA` input section.
- `.LUCITA` activates the LUCITA CI module (and the MOLTRA module to get 4-index transformed integrals). The specification of the LUCITA module input can be found in chapter 3.12. By default molecular orbitals with orbital energy between -10 au and +20 au are included, this can be modified in the `**MOLTRA` input section.

3.5.1 *DHFCAL — Hartree-Fock/Kohn-Sham calculation.

This section gives directives for closed shell and average-of-configurations Hartree-Fock calculations. Kohn-Sham calculations are performed by invoking the keyword `.DFT` under `**HAMILTONIAN`, see 3.4 for more details. **Note** that average-of-configurations Kohn-Sham calculations are not well defined.

1. Occupation

`.CLOSED SHELL` for each fermion irrep, give number of closed shell electrons

Arguments: Integers (NELEC(I), I=1, NFSYM)

Default: NELEC(1) = NELEC(2) = 0

`.OPEN SHELL` Specification of open shell(s). Specify the number of open shells:

Arguments: Integer NOPEN

For each open shell J the number of electrons and the number of active spinors, in the following format: NAELEC/ASC1,ASC2, where NAELEC is the number of active electrons and ASCx is the number of active spinors in corep x. Example:

```
.OPEN SHELL
1
5/0,6
```

This gives one open shell with five electrons in 6 spinors (= 3 Kramers pairs) in corep 2. Thus, the fractional occupation is 5/6.

The specification of the closed shell electrons is simple. For symmetry groups *without* inversion symmetry, there is only one fermion irrep, and you need only to specify the number of electrons. For example, the minimum wave function input for water (the C_{2v} point group), is

```
**WAVEFUNCTION
.DHF
*DHFCAL
.CLOSED SHELL
10
```

For point groups *with* inversion symmetry, you need to specify the distribution of the electrons in the two fermion irreducible co-representations (ircops) [22, 23]. For example, the minimum input for the Neon atom is

```
**WAVEFUNCTION
```

```
.DHF
*DHFAL
.CLOSED SHELL
4 6
```

The open shell module in DIRAC is based on average-of-configurations [24]. The simplest case is one electron in two spinors (= one Kramers pair). For this special case the average-of-configuration calculation gives the same result as the usual restricted open-shell Hartree-Fock. For all other cases the calculation gives the average energy of many states. An example of one electron in two spinors is the Boron atom

```
**WAVEFUNCTION
.DHF
*DHFAL
.CLOSED SHELL
4 0
.OPEN SHELL
1
1/0,2
```

The input is interpreted in the following way: We have 4 inactive electrons (in the 1s and 2s orbitals) and one open shell, where 1 electron is in two spinors (the $2p_{1/2}$ and $2\bar{p}_{1/2}$ spinors) in ircop 2. This calculation would give you the $P_{1/2}$ energy of Boron. We could also do a calculation with

```
**WAVEFUNCTION
.DHF
*DHFAL
.CLOSED SHELL
4 0
.OPEN SHELL
1
1/0,6
```

where we have the active electrons distributed in all six p -orbitals. This would give us the average energy of six states

$$\frac{2 \cdot E_{P_{1/2}} + 4 \cdot E_{P_{3/2}}}{6} \quad (3.1)$$

It is also possible to specify more than one open shell. Let us take a look at these two different inputs for the open shells of the Platinum atom ($6s^1 5d^9$)

```
(a) .OPEN SHELL
    2
    9/10,0
    1/2,0
(b) .OPEN SHELL
    1
    10/12,0
```

In the first example we get the average energy of 20 states: $\binom{10}{9}$. $\binom{2}{1} = 10 \cdot 2 = 20$. All 20 states are s^1d^9 states. In the second input we distribute the same ten electrons in all 12 spinors. This gives as total of $\binom{12}{10} = 66$ states, which are a mixture of one s^0d^{10} state, 20 s^1d^9 and 45 s^2d^8 states.

Note that the order of closed and open shells are assumed to be as in figure 3.1, that is the lowest molecular orbitals are doubly occupied, the next ones are occupied with the electrons of open shell no. 1, etc. Other orderings can be achieved by using the keywords `.REORDER MO'S` and overlap selection (`OVLSEL`) (see section 3.5.1).

To get the energies of the individual states present in the average-of-configurations, specify `.RESOLVE` (See sections 3.5 and 3.5.4. To get the energies of (some) of the individual states present in the average of configurations, you can use the `GOSCI` module (section 3.11), the `DIRRCI` module (section 3.10), or the `LUCITA` module (section 3.12).

`.AUTOCC` Program is allowed to change occupation during SCF cycles.

Default: Deactivated. However, the program will still try to do an automatic initial occupation if neither `.CLOSED SHELL ELECTRONS` nor `.OPEN SHELL ELECTRONS` is given.

2. Trial function

An SCF-calculation (HF or DFT) may be initiated in three different ways:

- using **MO-coefficients** from a previous calculation.
- using coefficients obtained by diagonalization of the one-electron Fock matrix: the **bare nucleus approach**.
- using **two-electron Fock matrix** from a previous calculation; this may be thought of as starting from a converged DHF potential

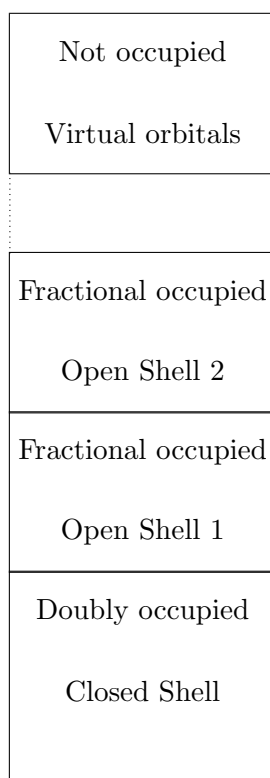


Figure 3.1: The order of closed and open shells.

Default is to start from MO-coefficients if the file `DFCOEF` is present. Otherwise the corrected bare nucleus approach is followed. In all three cases linear dependencies are removed in the zeroth iteration.

`.NOBNCR` Switch off the Bare Nucleus Correction. This correction is on per default and improves the screening of the nuclei by estimating two-electron repulsion via nuclear-attraction type integrals.

$$\langle X_A | \sum_C \frac{-Z_C \cdot \sum_j a_j e^{(-\alpha_j^C r_C^2)}}{r_C} | X_B \rangle, \quad X = L, S \quad (3.2)$$

The coefficients a_j and the exponents α_j^C in this expression are chosen according to Slater's rules to obtain an approximate atomic electronic density for the initial guess. For example, with one heavy element and without this correction (that is, with the bare nucleus Hamiltonian) all electrons will end up on that heavy element in the initial guess!

Default: `NOBNCR = .FALSE.`

`.TRIVEC` start SCF-iterations from vector file

`.TRIFCK` start SCF-iterations from two-electron Fock matrix from previous calculation (stored on file `DFFC2`).

3. Convergence criteria

Three different criteria for convergence may be chosen:

- the norm of the DIIS error vector $\mathbf{e} = [\mathbf{F}, \mathbf{D}]$ (in MO-basis). This corresponds to the norm of the electronic gradient and is the recommended convergence criterium. When you are only interested in the energy `EVCCNV = 1.0D-5` is usually sufficient. For properties and correlated methods you should converge to `EVCCNV = 1.0D-9`. The large positronic eigenvalues lead to a loss of precision that might lead to convergence problems. Remember also that a too tight screening threshold will hinder convergence, you should modify `.SCREEN` under `*TWOINT` if you modify `EVCCNV` or one of the other two convergence criteria.
- the difference in total energy between two consecutive iterations
- the largest absolute difference in the total Fock matrix between two consecutive iterations

The change in total energy is approximately the square of the largest element in the error vector or the largest change in the Fock matrix. *Default* is convergence on electronic gradient with threshold `SCFCNV = 1.0D-6`. Alternatively, the iterations will stop at the maximum number of iterations. Sometimes it may happen that the specified convergence criterium is too tight for the given basis set and/or other

input parameters. In this case one needs to decide whether one should proceed with post-HF steps (like correlation calculations) or not. The program decides this by looking at a secondary convergence criterium that gives the *allowed* convergence. This value is by default the same as first or *desired* convergence criterium but can be made lower to make sure that a calculation does not abort when the convergence is slightly above treshold.

.MAXITR maximum number of SCF - iterations
Arguments: Integer MAXITR
Default: MAXITR = 50

.EVCCNV converge on error vector (electronic gradient)
Arguments: Real SCFCNV(1) SCFCNV(2)

.ERGCNV threshold for convergence on total energy
Arguments: Real SCFCNV(1) SCFCNV(2)

.FCKCNV converge on largest absolute change in Fock matrix
Arguments: Real SCFCNV(1) SCFCNV(2)

Note that the secondary (*allowed*) threshold SCFCNV(2) may be omitted. It is then made equal to the first (*desired*) threshold SCFCNV(1).

4. Convergence acceleration

It is imperative to keep the number of SCF-iterations at a minimum. This may be achieved by convergence acceleration schemes.

- **Damping** The simplest scheme is damping of the Fock matrix that may remove oscillations. In iteration $n + 1$ the Fock matrix to be diagonalized is:

$$\mathbf{F}' = (1 - c)\mathbf{F}_{n+1} + c\mathbf{F}_n; \quad c - - - \text{damping factor} \quad (3.3)$$

- **DIIS** (Direct Inversion of iterative Subspaces [25, 26, 27]) may be thought of as generalized damping involving Fock matrices from many iterations. Damping factors are obtained by solving a simple matrix equation involving the B-matrix constructed from error vectors (approximate gradients). Linear dependent columns in the B-matrix is removed.

In DIRAC DIIS takes precedence over damping.

.DIISTH change default threshold for initiation of DIIS, based on largest element of error vector
Arguments: Real DIISTH — convergence threshold for initiation of DIIS
Default: A very large number.

- .MXDIIS maximum dimension of B-matrix in DIIS module
Arguments: Real MXDIIS — maximum dimension of B-matrix
Default: MXDIIS = 10
- .NODIIS do NOT perform Direct Inversion of Iterative Subspaces (DIIS)
Default: DIISMO is activated for closed-shell calculations, DIISAO is activated for average-of-configurations calculations.
- .DAMPFC change default damping factor
Arguments: Real DAMPFC — damping factor
Default: DAMPFC = 0.25.
- .NODAMP do NOT perform damping of Fock matrix
Default: Damping is activated, but DIIS takes precedence. In case all columns in the B-matrix is removed by linear dependency damping is activated.
- .DIISMO Activate DIIS in orthogonal basis (MO) with the error vector as described above ($\mathbf{e} = [\mathbf{F}, \mathbf{D}]$). This can only be used for closed-shell calculations.
- .DIISAO Activate DALTON -like DIIS using AO-basis. This implementation works both for closed and open shell calculations. The error vector is $\mathbf{e} = \mathbf{f} - \mathbf{f}^\dagger$ where \mathbf{f} is given by

$$\begin{aligned} \mathbf{f} = & \mathbf{C}^\dagger \cdot \mathbf{S}_{AO} \\ & \cdot \left(\mathbf{D}_{AO}^C \cdot \mathbf{F}_{AO}^D \right. \\ & \left. + \sum_{O \in O} f_O \cdot \mathbf{D}_{AO}^O \cdot \left(\mathbf{F}_{AO}^D + (a_O - 1) \mathbf{Q}_{AO}^{V,O} \right) \right) \mathbf{C} \end{aligned}$$

5. State selection

Convergence can be improved by selection of vectors based on overlap with vectors from a previous iteration. This method may also be used for convergence to some excited state.

If dynamic overlap selection is used, the vector set from the *previous* iteration is used as the criterium. For the first iteration either restart vectors or vectors generated by the *bare nucleus approach* (NOT recommended!) are used.

If .NODYNSEL is given either the restart vectors or the *bare nucleus* vectors are used, i.e. the overlap selection vectors are *not* updated in each iteration. Please note, that overlap selection based on vectors from the *bare nucleus approach* is not recommended.

Overlap selection is very useful together with the keyword .REORDER MO'S . This will reshuffle the vectors within the restart coefficients. Example: First one might do a

open shell calculation on Boron, this would give the $P_{1/2}$ state. But if we restart on the $P_{1/2}$ coefficients, interchange the $p_{1/2}$ with the $p_{3/2}$ orbitals, and request overlap selection, we can converge to the $P_{3/2}$ state. There also exists a keyword for reordering the converged DHF orbitals. This is useful for reordering the orbitals for the 4-index transformation and subsequent correlation calculations (CCSD, CI etc.) (see keyword `.POST DHF REORDER MO'S` .

- `.OVLSEL` activate dynamic overlap selection
Default: No overlap selection.
- `.NODYNSEL` No dynamic update of overlap selection vectors.
Default: Dynamic update
- `.REORDER MO'S` Interchange initial molecular orbitals. The start orbitals from DFCEOEF are read and reordered.
Argument(s): Character (A72)
For each fermion corep give the new order of orbitals.
Example:

```
.REORDER MO's
1 .8,10,9
```
- `.POST DHF REORDER MO'S` Interchange converged molecular orbitals. The orbitals from DFCEOEF are read and reordered just before exiting the DHF subroutine.
Argument(s): Character (A72)
For each fermion irrep give the new order of orbitals.
Example:

```
.POST DHF REORDER MO's
1 .8,10,9
```

6. Iteration speedup

The total run time may be reduced significantly by reducing the number of integrals to be processed in each iteration:

- **Screening on integrals:** Thresholds may be set to eliminate integrals below the threshold value [2]. The threshold for LL-integrals is set in the basis file, but this threshold may be adjusted for SL- and SS-integrals by threshold factors (set in `*INTEGRALS`):
 - Threshold for LL-integrals: `THRS`
 - Threshold for SL-integrals: `THRS*THRFACT(1)`
 - Threshold for SS-integrals: `THRS*THRFACT(2)`

These thresholds may be set in the integral section.

- **Screening on density:** In direct mode further reductions are obtained by screening on the density matrix as well [2]. This becomes even more effective if one employs **differential densities**, that is

$$\Delta\mathbf{D} = \mathbf{D}_{n+1} - \alpha \cdot \mathbf{D}_n \quad (3.4)$$

The default value for α is

$$\alpha = \frac{\mathbf{D}_{n+1} \cdot \mathbf{D}_n}{\mathbf{D}_n \cdot \mathbf{D}_n} \quad (3.5)$$

which corresponds to a Gram-Schmidt orthogonalization. As SCF converges α goes towards 1, but α can also explicitly be set equal to 1 with FIXDIF.

- **Neglect of integrals:** The number of integrals to be processed may be reduced even further by adding SL- and SS-integrals only at an advanced stage in the DHF-iterations, as determined either by the number of iterations or by energy convergence. The latter takes precedence over the former.

.NODSCF	do not perform SCF-iterations with differential density matrix <i>Default:</i> use differential density matrix in direct SCF.
.FIXDIF	Set α in Eq. (3.4) equal to 1. <i>Default:</i> FIXDIF = .FALSE.
.CNVINT	set threshold for convergence before adding SL- and SS-integrals to SCF-iterations. <i>Arguments:</i> Reals CNVINT(1)(SL),CNVINT(2)(SS) <i>Default:</i> Very large numbers.
.ITRINT	set number of iterations before adding SL- and SS-integrals to SCF-iterations. <i>Arguments:</i> Integers ITRINT(1) (SL),ITRINT(2)(SS) <i>Default:</i> ITRINT(1) = ITRINT(2) = 1
.INTFLG	specify what two-electron integrals to include <i>Arguments:</i> Integers ILL,ISL,ISS IXX = 1(on)/0(off) (XX = LL, SL, or SS) <i>Default:</i> INTFLG from **GENERAL .

7. Output control

.PRINT	general print level <i>Arguments:</i> Integer IPRSCF <i>Default:</i> IPRSCF = 0
--------	---

- .EIGPRI control printing of electron and positron solutions
Arguments: Integers IPREIG(1) (electron),IPREIG(2)(positron)
Default: Electronic eigenvalues printed.
- .PHCOEF phase adjustment of coefficients (on DFCOEF): make the largest element of a given orbital real and positive

8. Eliminating/freezing orbitals

In studies of electronic structure it may be of interest to eliminate or freeze certain orbitals. This option is furthermore useful for convergence, in particular to excited electronic states. A simple case is the thallium atom. The ground state $^2P_{1/2}$ has the electronic configuration $[\text{Xe}]4f^{14}5d^{10}6s^26p_{1/2}^1$. The first excited state $^2P_{3/2}$ with the electronic configuration $[\text{Xe}]4f^{14}5d^{10}6s^26p_{3/2}^1$ can easily be accessed by first calculating the ground state, then eliminating the $6p_{1/2}^1$ from the ensuing calculation. In the final calculation the excited state is relaxed using overlap selection. The use of frozen orbitals is demonstrated in test 33.frozen: When the geometry of the water molecule is optimized with the oxygen 1s and 2s orbitals frozen, a bond angle of 96.242 degrees is found, contrary to the 90° one might have expected when s-p hybridization is thus blocked [28].

The elimination of orbitals is achieved by projecting the selected orbitals out of the transformation matrix to orthonormal basis. The selected orbitals can be expressed either in the full molecular basis or in the basis set of the chosen fragment. In the latter case, the same set of fragment orbitals can in the case of atomic fragments be used at different molecular geometries. One may even perform a geometry optimization, but only using the numerical gradient. When freezing orbitals the selected orbitals are first eliminated from the transformation to orthonormal basis, but then reintroduced in the backtransformation step. They will appear in the output with zero orbital eigenvalues. Note that when freezing orbitals the orbitals to be eliminated must be specified as well. The frozen orbitals must be a subset of the eliminated orbitals.

Fragments are defined with respect to the list of symmetry-independent atoms appearing in the DIRAC basis file: Consider the water molecule in the full C_{2v} symmetry. Then there are two fragments: the oxygen atom and the H_2 moiety. However, with no symmetry there will be three fragments: the oxygen atom and the two hydrogen atoms. At the moment there are no orthonormalization of fragments on different fragments and so in practice one should only use orbitals from one fragment.

- .PROJEC eliminate orbitals by projecting them out of the transformation matrix to orthonormal basis
Arguments: Integer NPRJREF - number of fragments. Then for each

fragment read name PRJFIL of coefficient file followed by an orbital string (see section 3.13.1) of selected orbitals for each fermion irrep.

```
DO J = 1,NPRJREF
  READ(LUCMD,'(A6)') PRJFIL(J)
  READ(LUCMD,'(A72)') (VCPROJ(I,J),I=1,NFSYM)
ENDDO
```

- .PRJTHR - smallest norm accepted when eliminating orbitals
Arguments: Real PRJTHR — projection threshold
Default: PRJTHR = 1.0D-10.
- .OWNBAS - eliminated/frozen orbitals are given in the fragment basis. Note that the list of fragments is assumed to follow the list of symmetry independent nuclei in the DIRAC basis file.
- .FROZEN - freeze orbitals (must only be used in conjunction with the PROJEC keyword. For each fermion irrep, give an orbital string (see section 3.13.1) of orbitals to freeze.
Arguments: Character (A72) (VCFROZ(I),I=1,NFSYM)

3.5.2 Sample input

```
*DHFCAL
.PRINT
1
.NELECT
10 0
.MAXITR
30
.EVCCNV
1.0D-6
.MXDIIS
10
.CNVINT
1.0D-2 1.0D-4
.EIGPRI
1 0
```

3.5.3 *MP2CAL — MP2 calculation.

This section gives directives for the integral-direct closed-shell MP2 calculation.

Basic options

.OCCUPIED active occupied electronic solutions. For each fermion irrep, give an orbital string (see section 3.13.1) of active orbitals.

Arguments: Character (A72) (MP2_INDSTR(1,I), I=1,NFSYM)

Default: the occupied electronic solutions.

.VIRTUAL active virtual solutions. For each fermion irrep, give an orbital string (see section 3.13.1) of active orbitals.

Arguments: Character (A72) (MP2_INDSTR(2,I), I=1,NFSYM)

Default: all the unoccupied virtual solutions.

Advanced options

.INTFLG specify what two-electron integrals to include in the direct MP2 calculation.

Arguments: Integers ILL, ISL, ISS

IXX = 1(on)/0(off) (XX = LL, SL, or SS)

Default: INTFLG from **GENERAL .

.SCREEN Threshold for screening in 4-index transformation. A negative number deactivates screening.

Arguments: Real SCRMP2

Default: 1.0D-14

Programmers options

.PRINT print level

Arguments: Integer IPRMP2

Default: IPRMP2 = 0

.VIRTHR threshold for neglecting highlying virtual orbitals. Neglect all orbitals with energy above DMP2.VIRTHR. This option is retained for backwards compatibility, it is advised to use the more general energy option in the orbital string input.

Arguments: Real DMP2_VIRTHR

Default: No threshold

.IJTSK max. number of I (or J) orbitals (occupied orbitals) in each calculation batch. Usually one batch is sufficient but larger calculations may be possible by reducing the batch size. Reducing the batch size to half will approximately reduce the memory requirements to 1/4 and increase the CPU time by a

factor 4.

Arguments: Integer IJTSK

Default: All active occupied electronic solutions.

.SCLMEM Internal memory available for scalar untransformed integrals. Larger batches will be written to disk.

Arguments: Integer MAXSCL

Default: Is set by program and is usually sufficient.

Sample input for valence + subvalence Au₂ calculation. The 19 active occupied Kramers pairs are split in two and the calculation is done in three batches (1..10+1..10, 11..19+1..10, and 11..19+11..19 for I+J). All orbitals with energy above 100 a.u. is neglected in the virtual space. No SS integrals are included. Note that the orbitals in the active space need not be consecutive.

```
.NELECT
80 78
.....
*MP2CAL
.OCCUP
24,32..40
24,32..39
.VIRTUAL
all
all
.VIRTHR
100.0
.INTFLG
1 1 0
.IJTSK
10
```

3.5.4 *RESOLVE — Resolve open-shell states.

The individual electronic states of an average-of-configurations calculation are resolved by a full CI in the open-shell manifold(s) using the GOSCIP module. Note that even if this is a small CI one needs in principle to generate *all* AO-integrals for the initial 4-index transformation. The cost of calculation can be greatly reduced invoking integral screening (controlled by `.SCREEN`) in the 4-index transformation, possibly also leaving out at least SS-integrals (using `.INTFLG`).

.PRINT print level

- .INTFLG specify what two-electron integrals to include
Arguments: Integers ILL, ISL, ISS
 IXX = 1(on)/0(off) (XX = LL, SL, or SS)
Default: INTFLG from **GENERAL .
- .SCREEN Threshold for screening in 4-index transformation. A negative number deactivates screening.
Arguments: Real SCRRES
Default: 1.0D-14.
- .SCHEME Integral transformation algorithm
Arguments: Integer ISTRES
Default: ISTRES = 4

3.5.5 *MVOCAL — Modified virtual orbitals.

The virtual canonical Hartree-Fock orbitals is not the optimal choice for correlated calculations in which the virtual space is truncated; the lower orbitals tend to be diffuse since they see the $N + 1$ -electron system. One may exploit the freedom of rotation amongst the virtual orbitals to obtain orbitals better suited for correlated calculations. The default option in DIRAC is to generate the modified virtual orbitals by construction of a Fock operator for the system with some electrons removed, as first proposed by Bauschlicher [29]. The orbital string specified with .MVOVEC indicates what occupied orbitals to remove in the construction of the Hartree-Fock potential.

Note that the resulting modified virtual orbitals are no longer canonical and can *not* be used in MP2 calculations; they can be used in the DIRCI and RELCCSD modules.

- .PRINT print level
Arguments: Integer IPRMVO
Default: IPRMVO = 0
- .VECMVO Read orbital string (see section 3.13.1) of orbitals to be deleted from the Hartree-Fock potential when constructing cationic Fock operator
 READ(LUCMD, '(A72)') (VECMVO(I), I=1, NFSYM)
- .INTFLG specify what two-electron integrals to include during the construction of the modified virtual orbitals
Arguments: Integers ILL, ISL, ISS
 IXX = 1(on)/0(off) (XX = LL, SL, or SS)
Default: INTFLG from **GENERAL .

3.6 **ANALYZE — analyze wave function

This section allows for the analysis of the final wave function.

Basic options By default no analysis modules are activated.

- .DENSIT write density to formatted file in Gaussian cube format
- .MULPOP perform Mulliken population analysis
- .PRIVEC print vectors
- .PROJEC perform projection of vectors onto another vector set in the basis
- .RH01 punch density along the lines between centers

3.6.1 *PRIVEC — Vector print

Vectors read from DFCOEF may be printed.

Basic options

- .VECPRI for each fermion irrep, give an orbital string (see section 3.13.1) of orbitals to print.
Arguments: Character (A72) (VECPRI(I), I=1,NFSYM)
Default: the occupied electronic solutions.

Advanced options

- .PRICMP separate control of printing of large and small components
Arguments: Integers IPRCMP(1)(large),IPRCMP(2)(small).
Default: Only large components printed.
- .AOLAB print vectors in AO-basis
Default: print vectors in SO-basis

Programmers options

- .PRINT print level

3.6.2 *MULPOP — Mulliken population analysis

Mulliken population analysis is performed in AO-basis. The analysis is based on the concept of *labels*. Each basis function is labeled by its functional type and center (and boson irrep when in SO-basis, the default). The labels are given in output. A set of primitive labels may be collected to group labels as specified by the user.

.VECPOP for each fermion irrep, give an orbital string (see section 3.13.1) of orbitals to analyze.

Arguments: Character (A72) (VECPOP(I), I=1, NFSYM)

Default: the occupied electronic solutions.

Advanced options

.NETPOP give Mulliken net/overlap populations in addition to gross populations (default).

Default: Don't give net/overlap populations.

.LABDEF define labels for use in Mulliken population analysis

Arguments: Integer NPOPLAB - number of labels to define. Then read label name and orbital string (referring to labels, see section 3.13.1) for each fermion irrep.

```
DO I = 1, NPOPLAB
  READ(LUCMD, '(A12,A60)') POPLAB(I), LABPOP(I)
ENDDO
```

.AOLAB base definition of labels on AO-basis

Default: base definition of labels on SO-basis

.INDSML use individual small component labels

Default is to gather all small component functions belonging to a given center and irrep, irrespective of function type.

Programmers options

.PRINT print level

Arguments: Integer IPRPOP

Default: IPRPOP = 0

3.6.3 *PROJECTION — Projection analysis

The current solution may be projected down onto another set of coefficients generated from the basis, e.g. one may project molecular solutions down onto atomic solutions in order to evaluate atomic contributions. The coefficients of the fragments are read simultaneously and the overlap between them is taken into account [30]. The projection analysis can be thought of as a Mulliken analysis based on the atomic (fragment) orbitals; it is therefore very much less sensitive to the basis set.

Normally, the fragments are calculated in the full molecular basis. This is done by zeroing charges of the remaining atoms of the molecule in the basis set input and adjusting occupation in the menu file. It is, however, possible to calculate the fragments in the own basis (a subset of the full molecular basis) using the keyword `.OWNBAS`. The advantage is faster calculations and conservation of atomic symmetry for atomic fragments. To avoid working with symmetry-combinations of atomic centers for the fragments, it may sometimes be advantageous to dump molecular coefficients in C_1 symmetry using the keyword `.ACMOUT` (section 3.2) and do the analysis without symmetry. When the keyword `.OWNBAS` is used, one then needs to calculate each atomic type only once in its own basis in order to do the complete analysis.

Basic options

- `.VECPRJ` for each fermion irrep, give an orbital string (see section 3.13.1) of orbitals to analyze.
Arguments: Character (A72) (VECPRJ(I), I=1, NFSYM)
Default: the occupied electronic solutions.
- `.VECREF` first give number of fragments to project onto, and then for each fragment give filename of MO coefficients and for each fermion irrep, give an orbital string (see section 3.13.1) of reference orbitals.
Argument: Integer NREFS — number of fragments.
Arguments: Character (A72) (REFFIL(IFRAGMENT))
Arguments: Character (A72) (VECREF(I, IFRAGMENT), I=1, NFSYM)
Default: none, all lines must be given.
- `.OWNBAS` calculated fragments in their own basis. **NOTE!** This keyword must be used with some care as the list of fragments is assumed to be identical to that of symmetry independent centers.

Advanced options

.PROTHR set threshold for absolute value of projection coefficients to be printed

Default: PROTHR = 1.0D-2

Arguments: Real PROTHR

.WGPOP Split overlap densities according to weight of contributions

Default: Deactivated.

Programmers options

.PRINT print level

Arguments: Integer IPRPRJ

Default: IPRPRJ = 0

3.6.4 *DENSIT — generate density for visualization

The density can be dumped to formatted file in Gaussian cube format for subsequent visualization, e.g. using the MOLEKEL package (see <http://www.cscs.ch/molekel/>).

The Gaussian cube format stores volumetric data as well as atomic positions. The volume is defined by three axes and an origin.

The specific format is:

lines 1-2 Two comment lines

line 3 NATOM XO YO ZO
Number of atoms followed by origin of volumetric data.

lines 4-6 The next three lines give the number of voxels along each axis (x, y, z) followed by the axis vector. If the sign of the number of voxels in a dimension is positive then the units are Angstroms, if negative then atomic units are used.

NATOM lines For each atom a line of atomic number followed by coordinates.

The remaining file contains the volumetric data, in this case the density.

The large and small component density is written to files `rhoL.cube` and `rhoS.cube`, respectively. They can be recovered from the work area by the command

```
pam -incmo -get 'rhoL.cube rhoS.cube' ...
```

Note also that the coefficient file `DFCOEF` must be present in order to generate the density.

.PRINT print level

Arguments: Integer IPRRHO

Default: IPRRHO = 0

.DOCUBE specify large/small component density generation

Arguments: Integer ILC,ISC Values: 0(off),1(on).

Default: ILC = ISC = 1 meaning both large and small components density to be generated

.NCUBE number of voxels along the sides of the cube

Arguments: Integer NCUBE(1),NCUBE(2),NCUBE(3)

Default: NCUBE(1-3) = 80

.CUBADJ adjust volume of cube.

Arguments: Real CUBADJ(1),CUBADJ(2)

Default: CUBADJ(1)=4.0D0, CUBADJ(2)=8.0D0

DIRAC will look at atomic positions and then try to place the cube such that most density is included. The origin of the cube is determined by finding the minimum atomic position in the x-, y-and z-direction separately and then add CUBADJ(1). The length of the vectors spanning the cube (parallelepiped) are analogously defined by maximal atomic positions, with CUBADJ(2) added.

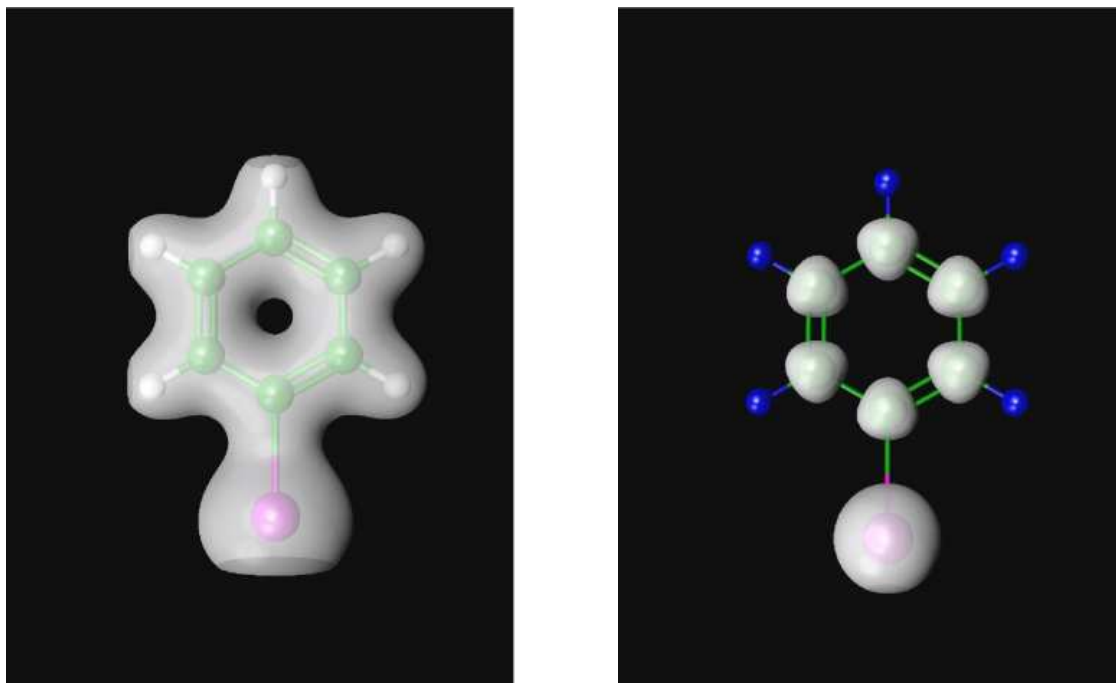


Figure 3.2: Large and small component density of iodobenzene plotted using the MOLEKEL software using cutoffs 0.01 and 0.0001, respectively.

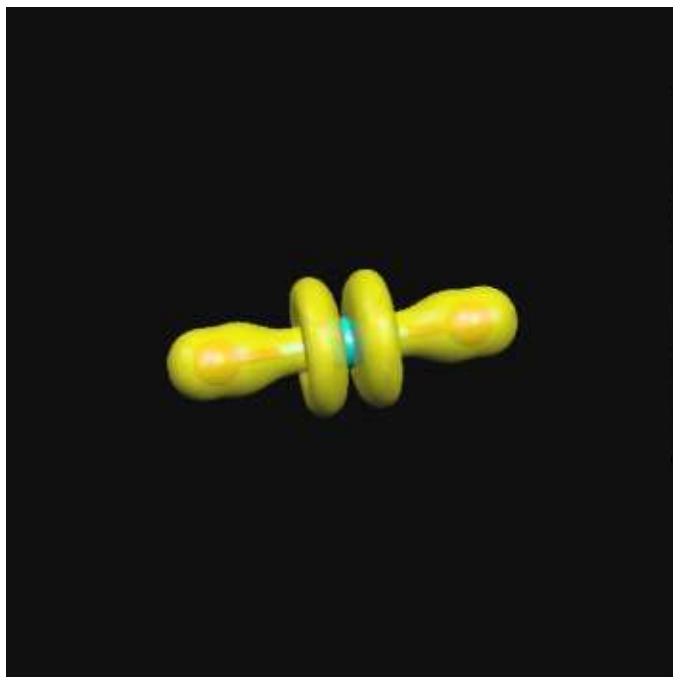


Figure 3.3: It is also possible to plot the density of an orbital by subtracting densities generated from a given coefficient file, varying the occupation. The figure shows the large component density of the HOMO of the uranyl cation obtained by subtracting from the density generated from regular occupation the density generated by reducing the occupation by two electrons.

3.6.5 *RH01 — Print density along bonds

For each unique pair (A,B) of centers a formatted file is created, containing four columns of numbers: The origin is placed at atom A, and the first column gives the coordinate in Angstroms along the line between centers A and B. The following columns contain the total, large and small component density, respectively (in atomic units). The file is named 'RH',NAMN(ICENTA),NAMN(ICENTB),IDEGB where ICENTB is the name of symmetry-independent center B and IDEGB the numbering of the dependent center. Underscores are inserted wherever there are blanks. The generation of the density requires the coefficient file DFCOEF. Note that one can in principle obtain the density along any line simply by introducing ghost centers. For uranyl illustrated here one may use the command

```
pam -incmo -get 'RHU__0___1 RHU__0___2' ...
```

```
.PRINT      print level
```

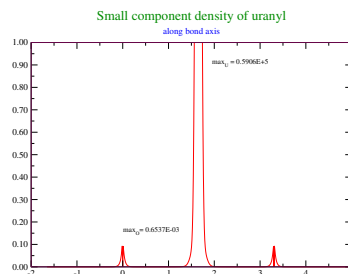
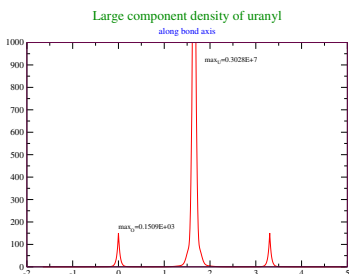
Arguments: Integer IPRH01

Default: IPRH01 = 0

.MESH step length for grid

Arguments: Real DSTEP

Default: DSTEP = 0.01D0



3.6.6 Sample inputs

An example of printing all orbitals in fermion irreducible co-representation 1 and some in irrep 2, large components only and using AO- and not the symmetry- adapted basis:

```
**ANALYZE
.PRIVEC
*PRIVEC
.VECPRI # what molecular orbitals to print
all
-4..2,4
.PRICMP # print only large components
1 0
.AOLAB # use AO-basis
```

An example of a Mulliken population analysis defining labels for each atomic shell and specifying orbital to analyze:

```
**ANALYZE
.MULPOP
*MULPOP
.LABDEF # redefinition of labels
3
H(s) 1
```

```
H(p)      2..4
H(d)      5..10
.VECPop   # what molecular orbitals to analyze
1..10
1,2
```

An example of projection analysis using two fragments, represented by coefficient files DFFRG1 and DFFRG2, respectively, and each calculated in their own basis:

```
**ANALYZE
..PROJEC
*PROJECTION
.OWNBAS   # use the individual basis set of fragments
.VECPRJ   # what molecular orbitals to analyze
1,2
1..3
.VECREf   # definition of reference orbitals
2
DFFRG1
all
all
DFFRG2
1..4
1..2
```

3.7 **PROPERTIES — property module

This section allows for the evaluation of a large number of molecular properties. Given that the necessary integral is present in the extensive integral menu of HERMIT, the property can in principle be calculated. Available properties include

- Expectation values (e.g. dipole moment and electric field gradients)
- Linear response properties (e.g. polarizability and NMR parameters).
- Quadratic response properties (e.g. hyperpolarizabilities).

For convenience some common properties can be specified directly in this section, which means that the user in principle does not need to know how they are calculated. Note, however, that response functions are by default static, but frequencies can be added in the relevant subsection (e.g. *LINEAR RESPONSE).

Properties which are not predefined must be specified in detail in the relevant input section (see 6.1).

By default no properties are calculated.

1. General control statements

- .PRINT print level
 Arguments: Integer IPRPRP
 Default: IPRPRP = 0
- .ABUNDANCIES For properties that make reference to isotopes, give threshold level
 (in % abundancy) for isotopes to print
 Arguments: Real ABUND
 Default: abundance greater than 1.0%

2. Predefined electric properties

- .DIPOLE evaluate electronic dipole moment (expectation value)
- .QUADRUPOLE evaluate electronic quadrupole moment (expectation value)
- .EFG evaluate electric field gradients (expectation value)
 Atomic centers may be restricted with .SELECT under **INTEGRALS .
- .NQCC evaluate nuclear quadrupole coupling constants (expectation value)
 Atomic centers may be restricted with .SELECT under **INTEGRALS.
- .POLARIZABILITY evaluate static electronic dipole polarizability (linear response function).

- .FIRST ORDER HYPERPOLARIZABILITY evaluate static electronic dipole first-order hyperpolarizabilities (quadratic response function). Results are also given for the static electronic dipole polarizabilities.
 - .VERDET evaluate Verdet constants (quadratic response function) for a dynamic electric field corresponding to Ruby laser wavelength of 694 nm and a static magnetic field along the propagation direction of the light beam (in this case, the default frequencies of the quadratic response function thus become $\omega_B = 0.0656$ and $\omega_C = 0$). A Verdet calculation cannot be specified in combination with other quadratic response calculations.
 - .TWO-PHOTON evaluate two-photon absorption cross sections (quadratic response function) at static frequencies. Input the number of desired states in each boson symmetry. Cannot be specified in combination with other quadratic response calculations.
- READ(LUCMD,*) (TPACNV(I),I=1,NBSYM)

3. Predefined magnetic properties

- .NMR evaluate nuclear magnetic shieldings and indirect spin-spin couplings (linear response functions)
Atomic centers may be restricted with .SELECT under **INTEGRALS .
- .SHIELDING evaluate nuclear magnetic shieldings (linear response functions). Print level (IPRPRP) 2 gives tensor and longer output. Print level 4 gives the raw values in symmetry coordinates as well.
Atomic centers may be restricted with .SELECT under **INTEGRALS .
- .SPIN-SPIN COUPLING evaluate indirect spin-spin couplings (linear response function)
Atomic centers may be restricted with .SELECT under **INTEGRALS.
- .DSO evaluate diamagnetic contribution to indirect spin-spin couplings as a expectation value
Atomic centers may be restricted with .SELECT under **INTEGRALS.
- .NSTDIAMAGNETIC evaluate diamagnetic contribution to nuclear magnetic shieldings as a expectation value
Atomic centers may be restricted with .SELECT under **INTEGRALS.

4. Other predefined properties

- .MOLGRD evaluate molecular gradient, i.e. $\frac{\partial E}{\partial X_A}$ where X_A are the coordinates of the nuclei. This is an expectation value of one- and two-electron operators. Normally the molecular gradient evaluation is not invoked

explicitly with this keyword but rather implicitly in the geometry optimization module.

.PVC Calculate matrix elements over the nuclear spin-independent parity-violating operator, e.g. calculate energy differences between enantiomers [31].

3.7.1 *EXPECTATION VALUE — Evaluate expectation values

This section gives directives for the calculation of expectation values.

Basic options

.OPERAT specification of general one-electron operators
see section 6.1 for details

Advanced options

.ORBANA analyze expectation value in terms of individual orbitals

Programmers options

.PRINT print level
Arguments: Integer IPREXP (default is zero).

3.7.2 *LINEAR RESPONSE — Linear response

Linear Response module written by T. Saue and H. J. Aa. Jensen [32]

The general form of a linear response function in the random phase approximation (RPA) is [32, 33]

$$\langle\langle A; B \rangle\rangle_{\omega} = -\mathbf{E}^{[1]\dagger}_A \left(\mathbf{E}^{[2]} - \omega \mathbf{S}^{[2]} \right)^{-1} \mathbf{E}^{[1]}_B \quad (3.6)$$

where $\mathbf{E}^{[1]}_A$ and $\mathbf{E}^{[1]}_B$ are property gradients corresponding to properties A and B, $\mathbf{E}^{[2]}$ is the molecular Hessian, $\mathbf{S}^{[2]}$ is the metric and ω is a frequency. The dimension of $\mathbf{E}^{[2]}$ does in general not allow an explicit inversion of the resolvent $\left(\mathbf{E}^{[2]} - \omega \mathbf{S}^{[2]} \right)^{-1}$, so one resorts to iterative techniques. We write

$$\left(\mathbf{E}^{[2]} - \omega \mathbf{S}^{[2]} \right)^{-1} \mathbf{E}^{[1]}_B = \mathbf{X} \quad (3.7)$$

By a slight rearrangement we arrive at the *linear response equations*

$$\left(\mathbf{E}^{[2]} - \omega\mathbf{S}^{[2]}\right) \mathbf{X} = \mathbf{E}_B^{[1]} \quad (3.8)$$

The RPA equation is solved by expanding \mathbf{X} in a set of trial vectors

$$\mathbf{X} = \sum_{i=1} \mathbf{b}_i a_i; \quad Y = \left[\mathbf{b}_1 \quad \mathbf{b}_2 \quad \dots \quad \mathbf{b}_n \right] \quad (3.9)$$

and solving the reduced equation

$$\left(\widetilde{\mathbf{E}}^{[2]} - \omega\widetilde{\mathbf{S}}^{[2]}\right) \mathbf{a} = \widetilde{\mathbf{E}}^{[1]}_B \quad (3.10)$$

in which

$$\widetilde{\mathbf{E}}^{[2]} = Y^\dagger \mathbf{E}^{[2]} Y; \quad \widetilde{\mathbf{S}}^{[2]} = Y^\dagger \mathbf{S}^{[2]} Y; \quad \widetilde{\mathbf{E}}^{[1]}_B = Y^\dagger \mathbf{E}_B^{[1]} \quad (3.11)$$

Keywords in this section are:

1. General control statements

.PRINT print level
Arguments: Integer IPXLR (default zero)

2. Definition of linear response function

.OPERAT specification of one-electron operators(both A and B)
 see section 6.1 for details

.A OPER specification of one-electron A operator
 see section 6.1 for details

.B OPER specification of one-electron B operator
 see section 6.1 for details

.TRIAB enforce triangularity of response function,
 that is $\langle\langle A; B \rangle\rangle_w \equiv \langle\langle B; A \rangle\rangle_w$; only one function calculated
Default: Deactivated.

.B FREQ specify frequencies of operator B
Default: Only static case.

 READ (LUCMD, *) NBFREQ
 READ (LUCMD,*) (BFREQ(J), J=1, NBFREQ)

.ALLCMB form all possible $\langle\langle A; B \rangle\rangle_w$ even if imaginary
Default: ALLCMB = .FALSE.

3. Control variational parameters

- .SKIPEE exclude all electron-electron rotations
Default: Include e-e rotations.
- .SKIPEP exclude all electron-positron rotations
Default: Include e-p rotations.

4. Control reduced equations

- .MAXITR maximum number of iterations
Arguments: Integer ITRXLR
Default: 50 iterations
- .MAXRED maximum dimension of matrix in reduced system
Arguments: Integer MAXRM
Default: MAXRM = 200
- .THRESH threshold for convergence of reduced system
Arguments: Real THCLR
Default: THCLR = 1.0D-5

5. Control integral contributions

The user is encouraged to experiment with these options since they may have an important effect on run time.

- .INTFLG - specify what two-electron integrals to include
Arguments: Integers ILL, ISL, ISS
 IXX = 1(on)/0(off) (XX = LL, SL, or SS)
Default: INTFLG from **GENERAL
- .CNVINT set threshold for convergence before adding SL- and SS-integrals.
Arguments: Reals CNVINT(1)(SL), CNVINT(2)(SS)
Default: Very large numbers.
- .ITRINT set number of iterations before adding SL- and SS-integrals.
Arguments: Integers ITRINT(1) (SL), ITRINT(2) (SS)
Default: ITRINT(1) = ITRINT(2) = 1

6. Control trial vectors

- .REAXVC read solution vectors from file XVCFIL
Arguments: String (A6) XVCFIL — file name
Default: No restart on solution vectors.

.XLRNRM normalize trial vectors. Using normalized trial vectors will reduce efficiency of screening.
Default: Use un-normalized vectors.

7. Advanced/debug flags

.ONLYSF only call FMOLI in sigmavector routine: only generate one-index transformed Fock matrix [32]

.ONLYSG only call GMOLI in sigmavector routine: only generate 2-electron Fock matrices using one-index transformed densities [32].

.STERNH set diagonal elements of orbital part of Hessian equal to $-2mc^2$ for $+-$ rotations, that is rotations between occupied positive-energy and virtual negative-energy orbitals.
Default: XSTERN = .FALSE.

.COMPRES - reduce number of orbital variation parameters by checking corresponding elements of gradient vector against a threshold. This may reduce memory.
Arguments: Real THRCOM
Default: THRCOM = 0.0D0 (no compression)

.NOPREC no preconditioning of initial trial vectors
Default: Preconditioning of trial vectors.

.RESFAC new trial vector will be generated only for variational parameter classes whose residual has a norm that is larger than a fraction $1/\text{RESFAC}$ of the maximum norm
Arguments: Real RESXLR
Default: RESXLR = 1000.0D0

3.7.3 *NMR — Controls evaluation of NMR parameters

This section gives directives for the calculation of NMR parameters.

Advanced options

.GAUGEO reads in a user defined gauge origin. Note that both the gauge origin and dipole origin are changed in order to have consistency.
Default: GAGORG = 0.0 0.0 0.0

```
READ (LUCMD, *) (GAGORG(ICOR), ICOR = 1, 3)
```

3.7.4 *MOLGRD — Controls evaluation of the molecular gradient

The section gives directives for control of a single point evaluation of the molecular gradient activated with `.MOLGRD`. Usually the molecular gradient module is called from within the geometry optimization module.

Advanced options

- `.TRICK` If activated the calculation of the SSSL and SSSS two-electron integral contributions to the gradient is skipped if their contribution is estimated to be small. An “empirical” estimate of the norms of the LS and SS two-electron gradients based on the norm of the LL two-electron gradient. If deactivated all contributions to gradient are calculated and if activated then only “necessary” contributions are calculated.
Default: Deactivated.
- `.INTFLG` - specify what two-electron integrals to include
Arguments: Integers ILL, ISL, ISS
IXX = 1(on)/0(off) (XX = LL, SL, or SS)
Default: INTFLG from **GENERAL

Programmers options

- `.PRINT` print level. A print level of less than 3 gives only the total gradient. Print levels from 3+ gives individual contributions to the gradient (kinetic energy gradient, nuclear attraction gradient etc.). Print levels of 5+ prints some matrices, and 10+ gives massive output!
Arguments: Integer IPRGRD
Default: IPRGRD = IPRGEN
- `.NUMGRA` - calculate the molecular gradient using numerical differentiation. *Default:* Analytical evaluation if implemented.

3.7.5 *QUADRATIC RESPONSE — Quadratic Response Functions

This section gives directives for the calculation of quadratic response functions [33]:

$$\langle\langle A; B, C \rangle\rangle_{\omega_1, \omega_2}$$

Basic options

.DIPLEN specification of dipole operators for A, B, and C
see section 6.1 for details

.A OPER specification of one-electron A operator
see section 6.1 for details

.B OPER specification of one-electron B operator
see section 6.1 for details

.C OPER specification of one-electron B operator
see section 6.1 for details

.B FREQ specify frequencies of operator B
Default: Only static case.

READ (LUCMD,*) NBQRFR
READ (LUCMD,*) (BQRFR(J), J=1, NBQRFR)

.C FREQ specify frequencies of operator C
Default: Only static case.

READ (LUCMD,*) NCQRFR
READ (LUCMD,*) (CQRFR(J), J=1, NCQRFR)

Advanced options

.ALLCMB evaluate all nonzero quadratic response functions and thereby disregarding
analysis of overall permutational symmetry.
Default: Evaluate only unique, nonzero, response functions.

.INTFLG specification of what two-electron integrals to include *Arguments:* Integers
ILL, ISL, ISS
IXX = 1(on)/0(off) (XX = LL, SL, or SS)
Default: INTFLG from **GENERAL

.XQRNRM normalize trial vectors. Using normalized trial vectors will reduce efficiency
of screening.
Default: Use un-normalized vectors.

.SKIPEE exclude all electron-electron rotations
Default: Include e-e rotations.

- .SKIPEP exclude all electron-positron rotations
Default: Include e-p rotations.
- .MAXITR maximum number of iterations
Arguments: Integer ITRXQR
Default: 30 iterations
- .MAXRED maximum dimension of matrix in reduced system
Arguments: Integer MAXQRM
Default: MAXQRM = 100
- .THRESH threshold for convergence of reduced system
Arguments: Real THCQR
Default: THCQR = 1.0D-5
- .ITRINT set number of iterations before adding SL- and SS-integrals.
Arguments: Integers ITRIQR(1) (SL),ITRIQR(2) (SS)
Default: ITRIQR(1) = ITRIQR(2) = 1
- .CNVINT set threshold for convergence before adding SL- and SS-integrals.
Arguments: Reals CNVXQR(1) (SL),CNVXQR(2) (SS)
Default: Very large numbers.

Programmers options

- .PRINT print level
Arguments: Integer IPREXP (default is zero).
- .NOPREC no preconditioning of initial trial vectors
Default: Preconditioning of trial vectors.
- .RESFAC new trial vector will be generated only for variational parameter classes whose residual has a norm that is larger than a fraction 1/RESFAC of the maximum norm
Arguments: Real RESXQR
Default: RESXQR = 1000.0D0

3.7.6 Input samples

Calculation of NMR parameters:

```
**PROPERTIES  
.NMR
```

Calculation of frequency-dependent polarizabilities, skipping rotations between occupied positive-energy and virtual negative-energy orbitals.

```
**PROPERTIES
.POLAR
*LINEAR RESPONSE
.B FREQ
3
0.0 0.1 0.2
.SKIPEP
```

3.8 **MOLTRA — integral transformation module

This module transforms integrals from the scalar AO-basis to integrals in the molecular spinor basis. The algorithm is not yet described in detail but it may be helpful to read [34] to find information about the double quaternion formalism that is used. The MOLTRA module is usually invoked by one of the correlation modules but can also be run as a stand-alone module by specifying `.MOLTRA` in the `**DIRAC` section of input. The input serves to change the default active space and to specify special wishes (e.g modification of `.INTFLG`). Three separate transformations are considered:

- Transformation of a core Fock matrix, giving effective one-electron matrices
- Transformation of two-electron Coulomb integrals
- Transformation of integrals over general (one-electron) operators

The output is written to the files `MDCINT`, `MRCONEE`, and `MDPROP`, respectively, and can be read in by the `RELCCSD` and `DIRRCI` programs to perform correlated calculations. Some common input needs to be specified both in this section and in `DIRRCI` (check `**DIRRCI` sections).

Basic options

`.ACTIVE` Specify the active set of spinors.
Arguments: One line with orbital strings (see 3.13.1) for each fermion irrep.
Default: energy -20.0 10.0 1.0.

Advanced options

`.INTFLG` specify what two-electron integrals to include in 2- and 4-index transformation. If you want different integral classes in the 2- and 4-index transformation, then use the `.INTFL2` and `.INTFL4` keywords below.

Arguments: Integers `ILL, ISL, ISS`
`IXX = 1(on)/0(off)` (`XX = LL, SL, or SS`).
Default: `INTFLG` from `**GENERAL`

`.INTFL2` specify what two-electron integrals to include in 2-index transformation (i.e., in the two-electron part of the Fock core matrix).

Arguments: Integers `ILL, ISL, ISS`
`IXX = 1(on)/0(off)` (`XX = LL, SL, or SS`).
Default: `INTFLG` from `**GENERAL`

- .INTFL4 specify what two-electron integrals to include in 4-index transformation.
Arguments: Integers ILL, ISL, ISS
IXX = 1(on)/0(off) (XX = LL, SL, or SS).
Default: INTFLG from **GENERAL
- .CORE specify the frozen core spinors
Arguments: One line with orbital strings for each fermion irrep
Default: all orbitals that are not active
- .POSITRONS Allow positronic spinors in active set. If this keyword is not specified then only electronic spinors will be included and any positronic spinors specified with the .ACTIVE keyword are ignored.
- .PRPTRA Perform transformation of property integrals.
Default: Skip transformation of property integrals.
- .NO2IND Skip 2-index transformation of effective Fock matrix
Default: Do not skip 2-index transformation.
- .NO4IND Skip 4-index transformation
Default: Do not skip 4-index transformation.
- .SCREEN Screening threshold in 4-index transformation
Default: SCREEN = 1.0D-14
(A negative value disables screening.)
- .THROUT Threshold for writing transformed 2-electron integrals to file
Default: THROUT = 1.E-14
- .RCORBS Recanonize orbitals before transforming
Default: RCORBS = .FALSE.

Programmers options

- .PRINT print level
Arguments: Integer IPRTRA
Default: IPRTRA = 0
- .4INDEX Ranges of active orbitals in 4-index transformation module specified for index 1 to 4 and fermion irrep 1 and 2. *Default:* ranges set by .ACTIVE keyword
- .2INDEX Ranges of active orbitals in 2-index transformation module specified for index 1 and 2 and fermion irrep 1 and 2. *Default:* ranges set by ACTIVE keyword

- .SCHEME Integral transformation algorithm
Default: SCHEME = 4
- .MDCINT Write 4-index transformed integrals to file
Default: MDCINT = .TRUE.
- .SCATTER Scatter MDCINT/4IND to all nodes
Default: SCATTER = .TRUE.
- .NOSCAT Do not Scatter MDCINT/4IND to all nodes
Default: Scatter to all nodes
- .PAR4BAS Specify number of batches for each node in 4-index transformation. Only working for parallel distributions.
Default: PAR4BAS = -1

3.8.1 *PRPTRA — Property integrals transformation

This section defines the property operators that should to be transformed. Such integrals are only used in an experimental section of RELCCSD , and this subsection can thus usually be omitted.

Programmers options

- .OPERAT specification of general one-electron operators
 see section 6.1 for details.
- .PRINT print level
Arguments: Integer IPRTRP
Default: IPRTRP = 0

3.8.2 Sample input

```
**MOLTRA
# use all spinors with energies between -12.0 and 20.0 au as active space
# note that one may NOT place a comment line directly after .ACTIVE
.ACTIVE
energy -12.0 20.0 1.0
```

3.9 RELCCSD — coupled cluster module

Relativistic Coupled Cluster module RELCCSD written by Lucas Visscher

This section specifies the input necessary for relativistic coupled cluster calculations. The program is currently capable of doing energy calculations at the MP2, CCSD and CCSD(T) levels of theory[35, 36], while first order properties are available only at the MP2 level of theory. Open shell states can be calculated via the Fock space Coupled Cluster method [37] in which electrons are added to or subtracted from a closed shell core. All sectors that involve no more than 2 replacements are available. The input should be given in namelist form and is subdivided in a general menu section and section describing energy, first order and second order property calculations.

Note that the integrals needed in CCSOPR must be specified in the ****MOLTRA** section.

3.9.1 RELCCSD — Generic and control input

Basic options

- DOENER calculate the energy
Default: DOENER = .TRUE.
- DOFOPR calculate the (relaxed) density matrix and first order properties
Default: DOFOPR = .FALSE.
- DOFSPC run a Fock space CC calculation
Default: DOFSPC = .FALSE.

Advanced options

- NELEC number of electrons in the two gerade and two ungerade irreps. This variable determines the reference determinant to be used in the exponential expansion of the wave function.
Arguments: Integer (NELEC(I), I=1, NFSYM*2)
Default: active electrons in these irreps (written by MOLTRA)
- NELEC_F1(2) number of electrons in the (un)gerade irreps of the Abelian symmetry group.
Arguments: Integer (NELEC_F1(I), I=1, number of g irreps)
Default: values set by MOLTRA or NELEC
- IPRNT set print level. Level 1 gives information on the convergence of the CCSD iterations and may be useful.
Default: Integer IPRNT = 0

TIMING write timing information
Default: TIMING = .FALSE.

The detailed NELEC_F1 and NELEC_F2 input is mainly of interest in atomic calculations where the program may use the $D_{\infty h}$ symmetry group. In that case this option permits precise specification of the reference determinant. Below we give an example for the high spin state of the nitrogen atom.

Programmers options

DEBUG write debug information
Default: DEBUG = .FALSE.

DOSORT Sort the incoming integrals over molecular spinors. This is a necessary step that can only be skipped in special (restart) cases.
Default: DOSORT = .TRUE.

NFROZ Number of frozen electrons in each irrep. Makes T1 and T2 amplitudes that concern frozen spinors zero. Only for test purposes.
Arguments: Integer (NFROZ(I), I=1,NFSYM*2)
Default: 0

3.9.2 CCENER — Input for energy module

Basic options

DOMP2 calculate the MP2 energy
Default: DOMP2 = .TRUE.

DOCCSD calculate the CCSD energy
Default: DOCCSD = .TRUE.

DOCCSDT calculate the CCSD(T) energy. In fact the program calculates three different non-iterative triple corrections (see [35]).
Default: DOCCSDT = .TRUE.

Advanced options

NTOL convergence tolerance of the DIIS error vector (10^{-NTOL})
Arguments: Integer NTOL
Default: NTOL = 12

- MAXIT maximum number of DIIS iterations to solve the CCSD equations
Arguments: Integer MAXIT
Default: MAXIT = 30
- MAXDIM maximum dimension of the reduced space
Arguments: Integer MAXDIM
Default: MAXDIM = 8

Programmers options

- NOCCS Ignore single excitations so that the CCD formalism is used. Only for test purposes.
Default: NOCCS = .FALSE.
- NOCCD Ignore double excitations so that the CCS formalism is used. Only for test purposes.
Default: NOCCD = .FALSE.

3.9.3 CCSORT — Input for sorting module

Basic options

- USEOE use orbital energies from the HF calculation instead of the diagonal elements of the reconstructed Fock matrix.
Default: USEOE = .TRUE.

3.9.4 CCFOPR — Input for first order properties

Only MP2 first order properties are currently available. They are computed by forming a relaxed density matrix, which is done (partly) in AO basis to obtain all relaxation contributions. The properties that are evaluated are those specified in the expectation value part of the input.

Basic options

- DOMP2G calculate the MP2 gradient, that is, expectation values using the relaxed one-electron density matrix. Note that this is not the molecular gradient, but other first order properties as specified in the expectation value part of the input.
Default: DOMP2G = .FALSE.

Advanced options

- NTOL convergence tolerance of the DIIS error vector (10^{-NTOL})
Arguments: Integer NTOL
Default: NTOL = 12
- MAXIT maximum number of iterations to solve the Z-vector equations
Arguments: Integer MAXIT
Default: MAXIT = 30
- MAXDIM maximum dimension of the reduced space
Arguments: Integer MAXDIM
Default: MAXDIM = 8

3.9.5 CCFSPC — Input for Fock space module**Basic options**

- DOEA calculate electron affinities
Default: DOEA = .FALSE.
- DOIE calculate ionization energies
Default: DOIE = .FALSE.
- DOEA2 calculate double electron affinities (add two electrons)
Default: DOEA2 = .FALSE.
- DOIE2 calculate double ionization energies (remove two electrons)
Default: DOIE2 = .FALSE.
- NACTP number of active spinors in which electrons may be put. This variable determines the active space for the electron affinities (EA) calculation.
Arguments: Integer (NACTP(I), I=1, NFSYM*2)
Default: no default : should always be set when using DOEA or DOEA2 !
- NACTH number of active spinors in which holes can be created. This variable determines the active space for the IE calculation.
Arguments: Integer (NACTH(I), I=1, NFSYM*2)
Default: no default : should always be set when using DOIE or DOIE2 !

Advanced options

- NTOL convergence tolerance of the DIIS error vector (10^{-NTOL})
Arguments: Integer NTOL
Default: NTOL = 12
- MAXIT maximum number of DIIS iterations to solve the CCSD equations
Arguments: Integer MAXIT
Default: MAXIT = 30
- MAXDIM maximum dimension of the reduced space
Arguments: Integer MAXDIM
Default: MAXDIM = 8

Programmers options

- FSSECT active Fock space sectors (ordered 00,01,10,11,02,20)
Default: FSSECT = 0,0,0,0,0,0

3.9.6 Sample input for RELCCSD

Sample input 1: Calculate CCSD(T) energy and MP2 first order properties. The full calculation is given as input example 1.energy+dipole in the test directory.

```
&RELCCSD TIMING=T, DOENER=T, DOFOPR=T &END
&CCFOPR DOMP2G=T &END
```

Sample input 2: Calculate CCSD energy for the nitrogen atom. We take $|\frac{1}{2}, \frac{1}{2}; \frac{3}{2}, \frac{1}{2}; \frac{3}{2}, \frac{3}{2}|$ (notation: $j_1, m_{j_1}; j_2, m_{j_2}$; etc.) as reference determinant.

```
&RELCCSD NELEC_F1=1,1, NELEC_F2=2,0,1,0 DOENER=T &END
&CCSORT USEOE=F &END
```

3.10 DIRRCI — direct CI module

Relativistic RASCI module written by Lucas Visscher

This section allows for relativistic configuration interaction calculations.

Note that when the namelist GOSCIP is also present in the input, GOSCIP will be called instead of DIRRCI !

The input should be given in namelist form.

3.10.1 RASORB — Specify the type of CI and the active spaces

Basic options

NELEC	Number of electrons (excluding frozen core electrons) <i>Default: NELEC = 0</i>
NRAS1	Number of spinors in the RAS1 space for each abelian irrep. <i>Default: NRAS1 = NSYMRP * 0</i>
NRAS2	Number of spinors in the RAS2 space for each abelian irrep. <i>Default: NRAS2 = NSYMRP * 0</i>
MAXH1	Maximum number of holes in RAS1 spinors <i>Default: MAXH1 = 0</i>
MAXE3	Maximum number of electrons in RAS3 spinors <i>Default: MAXE3 = 0</i>

3.10.2 CIRROOT — Select the state to converge on

Basic options

IREPNA	Abelian symmetry group of the desired state(s) <i>Default: First abelian symmetry in the list</i>
NROOTS	Number of states to optimize on <i>Default: NROOTS = 1</i>

Advanced options

ISTART	Start vector method 1. COSCI start vectors. 2. Determinant with lowest igenvalue.
--------	---

3. First (reference) determinant.
If the COSCI start vectors are not available then the default will be 3.

NSEL(NROOTS) Rank number of states to be optimized
Default: NSEL = 1,2,3,...

SELECT Select wave functions on basis of largest overlap with the start wave function
Default: SELECT = F

3.10.3 DIRECT — Convergence control

Basic options

CONVERE Convergence threshold for energy
Default: CONVERR = 1.0D-9

MAXITER Maximum number of direct CI iterations
Default: MAXITER = 10

Advanced options

CONVERR Convergence threshold for residual vector
Default: CONVERR = 1.0D-10

RESTART Restart on CI-vectors present in MRCFINV
Default: RESTART = F

CPUMAX Maximum amount of CPU-seconds to be used
Default: CPUMAX = 604800

3.10.4 OPTIM — Fine tuning of algorithm

Programmers options

IGENEX Write coupling coefficients to file (2) or calculates them when needed (1)
Default: IGENEX = 2

3.10.5 LEADDET — Analyze CI wave function

Advanced options

GETDET Get the list of dominant determinants
Default: GETDET = T

COMIN Print contributions of determinants only if the square of the coefficients is larger than COMIN.
Default: COMIN = 0.1

3.10.6 Sample input for DIRRCI

Sample input

```
**MOLTRA  
.ACTIVE  
2..5  
1..12  
*END OF
```

```
&RASORB  NELEC=7, NRAS1=1,1, NRAS2=2*0,3,3, MAXH1=2, MAXE3=2  &END  
&CIROOT  IREPNA=' 1Eu', NROOTS=3 &END  
&DIRECT  MAXITER=15 &END
```

3.11 GOSCIP — COSCI module

General Open Shell CI Program written by Olivier Visser

This module allows for small full configuration interaction calculations. It is invoked from within DIRRCI if the namelist GOSCIP is present in the input, or if `.RESOLVE` is specified when doing open shell HF calculations. The input should be given in namelist form².

3.11.1 GOSCIP — Specify the CI space

Basic options

NELEC Number of electrons (excluding frozen core electrons)
Default: NELEC = 0

Programmers options

IPRNT Print level
Default: IPRNT = 0

3.11.2 POPANA – analysis of wave function

Advanced options

THRESH print only determinants with coefficients higher than THRESH.
Default: 1.0D-3

DEGEN threshold for when several states are considered to be degenerate.
Default: 1.0D-10

SELPOP select only states with relative energies lower than SELPOP.
Default: 1.0D2

3.11.3 Sample input for GOSCIP

Sample input

```
&GOSCIP NELEC=5, IPRNT=1 &END
&POPANA THRESH=1.0D-4, DEGEN=1.0D-12, SELPOP=50.0 &END
```

²Historical note : This program was originally written for MOLFDIR [38] and was later also included in DIRAC.

3.12 LUCITA — direct GAS CI module

Spin-free relativistic GASCI module written by Jeppe Olsen, adaptation to DIRAC by Timo Fleig

LUCITA is a string-based Hamiltonian-direct general purpose configuration interaction (CI) program, based on LUCIA written by Jeppe Olsen (Theoretical Chemistry, Aarhus University, Denmark). It is capable of doing efficient CI computations at arbitrary excitation level, e.g. FCI, SDCI, RASCI, and MRCI using general active spaces. The code is interfaced [39] to molecular integrals obtained in the spin-free Dirac formalism and uses non-relativistic point group symmetry.

A central feature of the program is the Generalized Active Space (GAS) concept, in which the underlying total orbital space is subdivided into a basically arbitrary number (save for an upper limit) of subspaces with arbitrary occupation constraints. This is the most general approach to orbital space subdivisions. The program uses DIRAC orbitals from either a closed- or an open-shell calculation in a spin-free relativistic formalism or the non-relativistic Lévy-Leblond formalism (see section 3.4).

The technical limitations are roughly set by 100 million determinants in the CI expansion on a common PC, 400 million on larger PCs, and 1-2 billion on supercomputers with ample memory. For calculations involving more than 100.000 determinants and in particular higher than double excitations it is strongly recommended to use LUCITA in spin-free DIRAC applications.

If desired, the program also computes 1- and 2-particle densities from optimized CI wave functions. The density matrices may be printed along with natural orbital occupations and the corresponding eigenvectors (NOs).

LUCITA is provided with an interface section DIRLUC incorporated in the code. The DIRLUC environment reads the DIRAC input and converts it to a LUCIA input which is processed through an ASCII file. The types of CI which may be specified by keyword are: Full CI (FCI), CI with single and double excitations (SDCI), CI with single, double, triple, and quadruple excitations (SDTQ), restricted active space CI (RASCI), and generalized active space CI (GASCI). All types of CI can be considered special cases of a GASCI, so internally, DIRLUC transfers all information into GAS format.

3.12.1 General Input

Mandatory Keywords

INIWFC Initial HF wave function (closed- or open-shell)
 Values: DHFSCF or OSHSCF
 Default: none

CITYPE Type of CI calculation
Values: FCI, SDCI, SDTQ, RASCI, GASCI
Default: none

MULTIP State spin multiplicity
Default: none

Optional Keywords

TITLE One line with title of CI calculation
Default: `Default title`

NROOTS Number of states to optimize on
Default: `NROOTD = 1`

SYMMET State symmetry
Default: `ISSYMD = 1`

SZCALC Approximate size of calculation
Values: NOR for normal, LAR for large CI (.gt. 100 million determinants), HUG for huge CI
Default: `SZCALD = NOR`

INACTI Inactive orbitals per boson symmetry, separated by commas.
Default: All orbitals active.

DENSI Level of computed density matrices
Values: 1 for one-particle density only, 2 for one- and two-particle density matrices.
Default: `IDENSD = 0`

RSTRCI Optional restart from CI vector on file LUCVECT
Values: 1 restart calculation.
Default: `IRSTLT = 0`

PRINTL Local print flag for DIRLUC interface
Values: 0-10 range of flag
Default: `IPRNLD = 0`

PRINTG Global print flag for LUCIA
Values: 0-4 range. Use with care! Set to 1 for printing density matrices (if specified) and natural orbitals.
Default: `IPRNGD = 0`

3.12.2 Specific Input

GASCI

NACTEL Number of active electrons
Default: none

GASSHE Number and specification of GAS orbitals
 Number of GA spaces used
Values: 1-16 range.
Default: none
 Specification of GA spaces. One line per GAS with number of orbitals per
 boson symmetry, separated by commas.
Default: none

GASSPC Number and specification of sequential CI calculations
 Number of CI calculations with given GA spaces (currently only 1 is allowed!)
Default: none
 Specification of CI calculation. One line per GAS with 2 numbers each: The
 first gives the minimal number of accumulated electrons after this GAS, the
 second the corresponding maximum number, separated by blanks (defining
 occupation constraints of each GAS).
Default: none

RASCI

NACTEL Number of active electrons
Default: none

INACTI Inactive orbitals per boson symmetry, separated by commas.
Default: none

RAS1 RAS1 specification and maximum number of holes.
 Specification. Line with orbitals per boson symmetry, separated by commas.
Default: none
 Maximum number of holes in RAS1.
Default: none

RAS2 RAS2 specification
 Line with orbitals per boson symmetry, separated by commas.
Default: none

RAS3 RAS3 specification and maximum number of electrons.
 Specification. Line with orbitals per boson symmetry, separated by commas.
Default: none
 Maximum number of electrons in RAS3.
Default: none

3.12.3 Sample inputs for LUCITA

SDCI Sample input

```
*LUCITA
.INIWFC
  DHFSCF
.CITYPE
  SDCI
.MULTIP
  3
*END OF
```

GASCI Sample input

```
*LUCITA
.TITLE
  Aluminum atom in C2h symmetry, 7s5p2d (L) basis, SDT/SD CI (1s inactive).
.INIWFC
  OSHSCF
.CITYPE
  GASCI
.MULTIP
  2
.NACTEL
  13
.GASSHE
  5
  1,0,0,0   ! 1s
  2,0,0,0   ! 2s3s
  0,0,2,4   ! 2p3p
  5,2,1,2   ! 4s5s 4p 3d
  5,2,2,4   ! 6s7s 5p6p 4d
.GASSPC
  1
```

```
2 2
3 6
11 13
11 13
13 13
*END OF
```

3.13 Special features

3.13.1 Orbital strings

A subset of orbitals can be specified in two ways. Usually it suffices to specify the keyword “energy”, followed by three real numbers to specify a lower limit, upper limit and quasi-degeneracy threshold. All orbitals that have an energy between the two thresholds are then selected. The quasi-degeneracy threshold is used to extend the range if the lower or upper threshold cuts through a set of closely energy-spaced orbitals. It will move the upper or lower limit until it encounters a large enough gap between orbital energies. The default value used in the AO-MO integral transformation program to specify the active space for correlated calculations is e.g. given by the string

```
energy -10.0 20.0 1.0
```

which picks out all orbitals that have an energy between -10.0 and +20.0 au, with a minimum gap of 1.0 au. More detailed control is possible by using an orbital string. Here a character string is given where individual orbitals are listed separated by a comma or as a range of orbitals. The range limit is then separated by “..”. Positive energy (electronic) and negative energy (positronic) orbitals are indicated using positive and negative numbers, respectively. For instance the orbital string

```
-5..5,7
```

picks out the five upper positronic and five lower electronic orbitals plus electronic orbital number seven. If all orbitals, both electronic and positronic are wanted, then one may write

```
all
```

The concept of orbitals strings gives great flexibility in the specification of orbitals and serves as an alternative to editing of vector files. A drawback is, however, that one needs to count the number of orbitals manually, which may become cumbersome for cases with large ranges of virtual orbitals.

Chapter 4

DIRAC Basis File

The basis file defines the present basis set, the molecular geometry, and the symmetry of the system. The choice of an appropriate basis set is non-trivial in relativistic calculations because the available basis set sequences cover only part of the periodic table. For light elements (up to Argon) is recommended to use a standard nonrelativistic basis set and generate the small component basis by kinetic balance. For heavier elements the best choice would be to use one of the correlation consistent basis sets developed by K.G. Dyall (see in the basis directory for a more complete description). An alternative is to use the sets by K. Faegri that are also available in the basis set library but they may need to be extended by polarization functions. Both sets are to be used in uncontracted form in DIRAC . The small component basis set is generated automatically via the kinetic balance prescription, but can (by experienced users) also be specified explicitly. Do only try this if you have studied the theory and know what the criteria for small component basis sets are, there are many pitfalls that one needs to be aware of!

The detailed input is best described using an example.

4.1 Title and symmetry

```
1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that a tight p has NOT been added)
4:C 2 2 X Y
```

1 KEYWRD A6.

A keyword that is not used in DIRAC , it can be any sequence of characters.

2-3 TITLE(1), TITLE(2) (A72/A72).

Two arbitrary title lines.

```
4 CRT,NONTYP,KCHARG,SYMTXT,((KASYM(I,J),I=1,3),J=1,3), ID3, THRS
  (A1,I4,I3,A2,10A1,D10.2).
```

CRT must be set to 'C' to indicate that Cartesian Gaussians are to be used. If not, DIRAC will issue a warning and continue assuming Cartesian Gaussians. Note that by default transformation to spherical harmonic basis (modified for the small components) is done in the transformation to orthogonal basis.

NONTYP specify the number of atomic types, which is 2 in this example.

KCHARG The charge of the molecule. This will be used to determine initial Hartree-Fock/Kohn-Sham occupation, if it is not explicitly specified. A zero is assumed if no number is given here.

SYMTXT specify the number of generators of the symmetry group. For C_{2v} there are two generators, in this example σ_x and σ_y . You must specify explicitly a zero "0" if you want to run the calculation without symmetry. If SYMTXT is left blank, then DIRAC will analyze the geometry and find the appropriate symmetry group. This is also mandatory for linear molecules, where the full symmetry group $D_{\infty h}$ or $C_{\infty v}$ can be employed in some of the modules.

KASYM(I,J) Symmetry is restricted to the binary groups, that is D_{2h} and subgroups, which means that a symmetry operation acting on the main axes (x,y,z) will at most reverse their direction. A group generator is therefore identified by a 3-character string that specifies the axes reversed under its operation. Examples are given for the eight binary groups in Tab. 4.1. In this example the X and Y are given to indicate the generators σ_x and σ_y .

ID3 If not blank then the molecular coordinates are read in Ångströms instead of Bohrs (atomic units).

THRS Threshold for neglect of final integrals. Default is 1.0D-15, which will be used if no input is given here. A threshold of 1.0D-15 will give integrals correct to approximately 1.0D-13. Separate thresholds for SL- and SS-integrals may be specified in the menu file (see keyword THRFAC in section 3.3.3).

Table 4.1: Example definitions of the binary groups (- indicates blank character)

Group	SYMTXT	KASYM	Operations
D_{2h}	3	--Z--Y--X	$\sigma(xy), \sigma(xz), \sigma(yz)$
D_2	2	XY--YZ---	$C_2(z), C_2(x)$
C_{2v}	2	-Y-X-----	$\sigma(xz), \sigma(yz)$
C_{2h}	2	--ZXYZ---	$\sigma(xy), i$
C_2	1	XY-----	$C_2(z)$
C_s	1	--Z-----	$\sigma(xy)$
C_i	1	XYZ-----	i
C_1	0	-----	

4.2 Atomic coordinates

```

1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that a tight p has NOT been added)
4:C  2    2  X  Y
5:      8.    1
6:0      .0000000000      0.0000000000      -.2249058930

```

5 Q,NONT(I),QEXP (1X,F9.0,I5,F20.5)

Q The charge of this atom type. In the case of oxygen it is 8.

NONT Number of symmetry-distinct atoms of this type, which is 1 for oxygen in water.

QEXP Gaussian exponent for nuclear charge distribution (if blank, then the default value is used).

6 NAMN,(CORD(J),J=1,3) (A4,*)

NAMN Atom name. It is recommended to use a different name for each atom of the same type in order to be able to identify them in the output.

CORD The x -, y -, and z -coordinate in Bohrs (atomic units). ID3 on card 4 (see section 4.1) must be non-blank if you want to enter the coordinates in Ångströms. The Cartesian coordinates may be given in free format. However, *the name of the atom must still be left four places*, and no coordinates must enter the four first positions.

4.3 Large component basis set

The specification of the large component basis set follows the atomic coordinates. There are various possibilities for giving the basis set.

4.3.1 Basis sets from the basis set library

```

1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that this is a nonrelativistic basis)
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000    0.0000000000    -.2249058930
7:LARGE BASIS aug-cc-pVDZ

```

7 BSET,BSKEYWORD,BASFIL (A5,1X,A5,1X,A)

BSET Must be LARGE.

BSKEYWORD Must be BASIS when basis sets from the basis set library is requested.

BASFIL Name of the basis set.

4.3.2 Explicitly typed basis set

```

1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that a tight p has NOT been added)
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000    0.0000000000    -.2249058930
7:LARGE INTGRL  3    1    1    1
8:# S-TYPE FUNCTIONS
9:f  10    4
10: 11720.0000000  0.00071000 -0.00016000  0.00000000  0.00000000
11:  1759.0000000  0.00547000 -0.00126300  0.00000000  0.00000000
12:   400.8000000  0.02783700 -0.00626700  0.00000000  0.00000000
13:   113.7000000  0.10480000 -0.02571600  0.00000000  0.00000000
14:    37.0300000  0.28306200 -0.07092400  0.00000000  0.00000000
15:    13.2700000  0.44871900 -0.16541100  0.00000000  0.00000000
16:     5.0250000  0.27095200 -0.11695500  0.00000000  0.00000000

```

```

17:      1.0130000  0.01545800  0.55736800  0.00000000  0.00000000
18:      0.3023000 -0.00258500  0.57275900  1.00000000  0.00000000
19:      0.0789600  0.00000000  0.00000000  0.00000000  1.00000000
20:# P-TYPE FUNCTIONS
21:f   5   3
22:     17.7000000  0.04301800  0.00000000  0.00000000
23:      3.8540000  0.22891300  0.00000000  0.00000000
24:      1.0460000  0.50872800  0.00000000  0.00000000
25:      0.2753000  0.46053100  1.00000000  0.00000000
26:      0.0685600  0.00000000  0.00000000  1.00000000
27:# D-TYPE FUNCTIONS
28:f   2   2
29:      1.1850000  1.00000000  0.00000000  0.00000000
30:      0.3320000  0.00000000  1.00000000  0.00000000

```

7 BSET,BSKEYWORD,IQM,(JCO(K),K=1,IQM) (A5,1X,A6,12I5)

For backwards compatibility the keyword BSKEYWORD does not have to be present:

```
7:LARGE   3   1   1   1
```

is which case it is read as BSET,IQM,(JCO(K),K=1,IQM) (A5,12I5).

BSET Must be LARGE.

BSKEYWORD Must be INTGRL or EXPLICIT for explicitly typed basis sets.

IQM highest angular quantum number l plus one, e.g. s(1),p(2) etc. In this case it is 3, since we're using a *spd* basis set.

JCO number of blocks for each l -value. The memory requirements grow rapidly with the number of basis functions in a block (note for instance that four g functions actually are 60 basis functions, as there are 15 cartesian components of each g function). Memory requirements can therefore be reduced by splitting basis functions of the quantum number into different blocks. This will, however, decrease the performance of the integral calculation.

8,20,27 Lines starting with either !, \\$, or # are comments.

9,21,28 FRMT,NUCIJ,NRCIJ,ISGEN (A1,I4,2I5)

FRMT A single character describing the input format of the basis set in this block. The default format is (8F10.4) which will be used if FRMT is left blank. In this

format the first column is the orbital exponent and the seven last columns are contraction coefficients. If no numbers are given, a zero is assumed. If more than 7 contracted functions occur in a given block, the contraction coefficients may be continued on the next line, but the first column (where the orbital exponents are given) must then be left blank.

An **F** or **f** in the first position will indicate that the input is in free format. This will of course require that all contraction coefficients need to be typed in, as all numbers need to be present on each line. However, note that this options is particularly handy together with completely decontracted basis sets, as described below. Note that the program reads the free format input from an internal file that is 80 charcters long, and no line should therefore exceed 80 characters.

One may also give the format **H** or **h**. This corresponds to high precision format (4F20.8), where the first column again is reserved for the orbital exponents, and the three next columns are designated to the contraction coefficients. If no number is given, a zero is assumed. If there are more than three contracted orbitals in a given block, the contraction coefficients may be continued on the next line, though keeping the column of the orbital exponents blank.

NUCIJ Number of primitive Gaussians in this block.

NRCIJ Number of contracted Gaussians in this block. If a zero is given, an uncontracted basis set will be assumed, and only orbital exponents need to be given.

ISGEN specification of how to generate small component functions by kinetic balance:

ISGEN = 1 Small component functions generated upwards, e.g. $p \rightarrow d$

ISGEN = 2 Small component functions generated downwards, e.g. $p \rightarrow s$

ISGEN = 0, ISGEN = 3 Small component functions generated both upwards and downwards, e.g. $p \rightarrow s, d$

other values No small components functions generated.

This information is only used if the small component basis functions are to be generated by kinetic balance. If no number is given, **ISGEN = 0** is assumed. The routine that generates the small component basis set by kinetic balance will delete duplicate functions.

10-19,22-26,29-30 Exponents (first column) and contraction coefficients read. For an uncontracted basis set only the exponents are read. The format is given by **FRMT** above.

4.3.3 MOLFDIR-type basis sets

```

1:DIRAC
2:Water
3:using MOLFDIR basis for large component
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000    0.0000000000    -.2249058930
7:LARGE MOLFBAS Oxygen-xyz.bas

```

7 BSET,BSKEYWORD,BASFIL (A5,1X,A7,1X,A)

BSET Must be LARGE.

BSKEYWORD Must be MOLFBAS for MOLFDIR -type basis sets.

BASFIL Filename of the basis set. This file must be copied to the scratch area, for example using the pam script:

```
pam ... -copy H.bas ...
```

4.3.4 Even-tempered basis sets (geometric progressions)

The exponents are generated in an even-tempered series

$$\eta_{N-k+1} = \alpha\beta^{N-1}, \quad k = 1, \dots, N \quad (4.1)$$

```

1:DIRAC
2:Water
3:with even tempered basis set
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000    0.0000000000    -.2249058930
7:LARGE EVENTEMP 0.05 3.0 11 3 2 1 1
8:1..5
9:6..11
10:7..11
11:9..10

```

7 BSET,BSKEYWORD,ALPHA,BETA,N,IQM,(JCO(K),K=1,IQM) (A5,1X,A8,1X,*)

BSET Must be LARGE.

BSKEYWORD Must be EVENTEMP or GEOM even tempered basis sets..

ALPHA The parameter α in the even tempered series.

BETA The parameter β in the even tempered series.

N The parameter N in the even tempered series.

IQM highest angular quantum number l plus one, e.g. s(1),p(2) etc. In this case it is 3, since we're using a *spd* basis set for the large components.

JCO number of blocks for each l -value. The memory requirements grow rapidly with the number of basis functions in a block (note for instance that four g functions actually are 60 basis functions, as there are 15 cartesian components of each g function). Memory requirements can therefore be reduced by splitting basis functions of the quantum number into different blocks. However, this will decrease the performance of the integral calculation.

8-11 For each block the range of exponent used for that particular block is given. In this case the first block (s functions) will consist of exponents 1 – 5, the second block (s functions) of exponents 6 – 11, the third block (p functions) of exponents 7 – 11, and the last block (d functions) of exponents 9 – 10.

4.3.5 Well-tempered basis sets

The exponents are generated in an well-tempered series

$$\eta_N = \alpha \quad (4.2)$$

$$\eta_{N-k+1} = \eta_{N-k+2}\beta \left[1 + \gamma \left(\frac{k}{N} \right)^\delta \right], \quad k = 2, \dots, N \quad (4.3)$$

```

1:DIRAC
2:Water
3:with well-tempered basis set
4:C 2 2 X Y
5: 8. 1
6:0 .0000000000 0.0000000000 -.2249058930
7:LARGE WELLTEMP 0.05 2.5 2.0 6.0 11 3 2 1 1
8:1..5
9:6..11
10:7..11
11:9..10

7 BSET,BSKEYWORD,ALPHA,BETA,GAMMA,DELTA,N,IQM,(JCO(K),K=1,IQM)(A5,1X,A8,1X,*)

BSET Must be LARGE.
```

BSKEYWORD Must be WELLTEMP for well tempered series.

ALPHA The parameter α in the well tempered series.

BETA The parameter β in the well tempered series.

GAMMA The parameter γ in the well tempered series.

DELTA The parameter δ in the well tempered series.

N The parameter N in the well tempered series.

IQM highest angular quantum number l plus one, e.g. s(1),p(2) etc. In this case it is 3, since we're using a *spd* basis set.

JCO number of blocks for each l -value. The memory requirements grow rapidly with the number of basis functions in a block (note for instance that four g functions actually are 60 basis functions, as there are 15 cartesian components of each g function). Memory requirements can therefore be reduced by splitting basis functions of the quantum number into different blocks. However, this will decrease the performance of the integral calculation.

8-11 For each block the range of exponent used for that particular block is given. In this case the first block (s functions) will consist of exponents 1 – 5, the second block (s functions) of exponents 6 – 11, the third block (p functions) of exponents 7 – 11, and the last block (d functions) of exponents 9 – 10.

4.3.6 Family basis sets

Input for basis sets where the same set of exponents are used for all functions. This is analogous to the well and even tempered basis sets except that the exponents are not calculated from a formula, but must be given in the file. These exponents may come from a basis set optimization with GRASP [40].

```

1:DIRAC
2:Water
3:with family basis set
4:C 2 2 X Y
5: 8. 1
6:0 .0000000000 0.0000000000 -.2249058930
7:LARGE FAMILY 10 3 1 1 1
8: 6665.0000000
9: 1000.0000000
10: 228.0000000
11: 64.7100000
```

```

12: 21.060000
13: 7.4950000
14: 2.7970000
15: 0.5215000
16: 0.1596000
17: 0.0469000
18: 1..10
19: 6..10
20: 8..9

```

7 BSET,BSKEYWORD,N,IQM,(JCO(K),K=1,IQM) (A5,1X,A6,12I5)

BSET Must be LARGE

BSKEYWORD Must be FAMILY for a family basis sets.

N The number of exponents to be read.

IQM highest angular quantum number l plus one, e.g. s(1),p(2) etc. In this case it is 3, since we're using a *spd* basis set.

JCO number of blocks for each l -value. The memory requirements grow rapidly with the number of basis functions in a block (note for instance that four g functions actually are 60 basis functions, as there are 15 cartesian components of each g function). Memory requirements can therefore be reduced by splitting basis functions of the quantum number into different blocks. However, this will decrease the performance of the integral calculation.

8-17 Read N lines with exponents. The exponents are read in free format.

18-20 For each block the range of exponent used for that particular block is given. In this case the first block (s functions) will consist of exponents 1 – 10, the second block (p functions) of exponents 6 – 10, the last block (d functions) of exponents 8 – 9.

4.3.7 Dual family basis sets

A basis set analogous to the family basis set, except one set of exponents are used for all $l = 0, 2, 4, \dots$ basis functions (s, d, g, \dots) and another set is used for all $l = 1, 3, 5, \dots$ functions (p, f, h, \dots).

```

1:DIRAC
2:Water
3:with dual family basis set

```

```

4:C  2  2 X Y
5:      8.  1
6:0  .0000000000  0.0000000000  -.2249058930
7:LARGE DUALFAMILY  10  5  3  1  1  1
8:# s, d, g, ... exponents
9: 6665.0000000
10: 1000.0000000
11:  228.0000000
12:   64.7100000
13:   21.0600000
14:    7.4950000
15:    2.7970000
16:    0.5215000
17:    0.1596000
18:    0.0469000
19:# p, f, h, ... exponents
20:  9.4390000
21:  2.0020000
22:  0.5456000
23:  0.1517000
24:  0.0404100
25:# ranges ...
26: 1..10
27: 1..5
28: 8..9

```

```
7 BSET,BSKEYWORD,N1,N2,IQM,(JCO(K),K=1,IQM) (A5,1X,A10,14I5)
```

BSET Must be **LARGE**

BSKEYWORD Must be **DUALFAMILY** for a dual family basis sets.

N1 The number of exponents in set 1.

N2 The number of exponents in set 2.

IQM highest angular quantum number l plus one, e.g. $s(1),p(2)$ etc. In this case it is 3, since we're using a *spd* basis set.

JCO number of blocks for each l -value. The memory requirements grow rapidly with the number of basis functions in a block (note for instance that four g functions actually are 60 basis functions, as there are 15 cartesian components of each g function). Memory requirements can therefore be reduced by splitting

basis functions of the quantum number into different blocks. However, this will decrease the performance of the integral calculation.

- 9-18** Read N1 lines with exponents for the s, d, g, \dots basis set functions. The exponents are read in free format.
- 20-24** Read N2 lines with exponents for the p, f, h, \dots basis set functions. The exponents are read in free format.
- 26-28** For each block the range of exponent used for that particular block is given. In this case the first block (s functions) will consist of exponents 1 – 10 from set 1, the second block (p functions) of exponents 1 – 5 from set 2, the last block (d functions) of exponents 8 – 9 from set 1.

4.4 Small component basis set

After the large component basis set the small component basis set *can* be specified. If nothing is specified it is equivalent to specifying SMALL KINBAL (see below). There are three possibilities for giving the basis set.

4.4.1 By kinetic balance

```

1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that a tight p has NOT been added)
4:C 2 2 X Y
5: 8. 1
6:0 .0000000000 0.0000000000 -.2249058930
...
29: 1.1850000 1.00000000 0.00000000 0.00000000
30: 0.3320000 0.00000000 1.00000000 0.00000000
31:SMALL KINBAL

31 BSET,BSKEYWORD (A5,1X,A6)

```

Must be SMALL and KINBAL when requesting kinetic balance.

The small component basis is generating using kinetic balance. The small component basis set will *always* be uncontracted, independent of whether the large component basis set is contracted or not.

4.4.2 Explicitly typed basis set

```

1:DIRAC
2:Water
3:aug-cc-pVDZ basis (note that a tight p has NOT been added)
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000      0.0000000000      -.2249058930
...
29:      1.1850000  1.00000000  0.00000000  0.00000000
30:      0.3320000  0.00000000  1.00000000  0.00000000
31:SMALL INTGRL  3    1    1    1
32:...
33:...

```

31 BSET,BSKEYWORD,IQM,(JCO(K),K=1,IQM) (A5,1X,A6,12I5)

Analogous to LARGE INTGRL ... described above. BSKEYWORD can be INTGRL, EXPLICIT, or nothing.

4.4.3 MOLFDIR-type basis sets

```

1:DIRAC
2:Water
3:with MOLFDIR basis set for both large and small components
4:C  2    2  X  Y
5:      8.    1
6:0    .0000000000      0.0000000000      -.2249058930
7:LARGE MOLFBAS Oxygen-xyz.bas
8:SMALL MOLFBAS Oxygen-xyz.bas

```

8 BSET,BSKEYWORD,BASFIL (A5,1X,A7,1X,A)

Analogous to LARGE MOLFBAS ... described above.

Chapter 5

Output Files

A formatted output file is connected to DIRAC through standard output. The user may modify the output by setting various print levels in the menu file. DIRAC will in addition produce a formatted file `DFCYCL` containing a summary of the SCF process. In addition to the formatted DIRAC produces a number of unformatted files. They are

- **Control files**

<code>DFDIIS</code>	information about DIIS process
<code>DFEVEC</code>	direct access file with DIIS error vectors

- **Orbital coefficients**

<code>DFCOEF</code>	MO-coefficients from current SCF-iteration
---------------------	--

- **One-electron integrals and matrices**

<code>DF1INT</code>	one-electron integrals contributing to one-electron Fock matrix
<code>DFOVLP</code>	overlap matrices
<code>DFTMAT</code>	MO-transformation matrix
<code>DFFCK1</code>	One-electron Fock matrix (in QO basis)

- **Two-electron integrals and matrices**

For each integral class ($XX = LL, SL, SS$) in conventional mode:

<code>DFXXSA</code>	sorted singlet integrals (both Coulomb and exchange contributions)
<code>DFXXSB</code>	sorted singlet integrals (both Coulomb and exchange contributions)
<code>DFXXTA</code>	sorted triplet integrals (only exchange contributions)
<code>DFXXTB</code>	sorted triplet integrals (only exchange contributions)

DFTWXX scratch file of unsorted integrals from HERMIT
DFXXTB scratch file used in sorting process

In addition:

DFFCK2 two-electron Fock matrix in QO-basis

5.1 Restart

DIRAC has some restart facilities:

- When calculating a **new point** on a potential surface, DIRAC can start from the coefficients (the file `DFCOEF`), from the two-electron Fock matrix in AO-basis (the file `DFFCK2`) or from solutions of the one-electron Fock matrix (bare nucleus approximation). Default is to start from coefficients if the unformatted vector file `DFCOEF` is present; otherwise DIRAC uses the bare nucleus approximation. Restart on Fock matrix may be specified by the keyword `TRIFCK`.
- When restarting on the **same point** on the potential surface, DIRAC needs the formatted file `DFCYCL` to update status of the SCF process. The full SCF summary will be provided at the end of the current iteration, so that the output file from the previous SCF iterations is generally not needed. In addition DIRAC needs the coefficients (file `DFCOEF`). To restart on DIIS, DIRAC needs the following files: `DFDIIS`, `DFCMOS`, `DFFOCK` and `DFEVEC`. If DIIS is not requested, DIRAC may restart on damping if the file `DFFOCK` is present.

Chapter 6

Specification of One-Electron Operators

6.1 Syntax for specification of operators

A general property operator in 4-component calculations is generated from linear combinations of the basic form

$$a_i \mathbf{M}_i \Omega_j \tag{6.1}$$

where \mathbf{M}_i is one of the following 4×4 -matrices:

$$\begin{aligned} \mathbf{M}_i = & I_4, \gamma_5, \\ & \alpha_x, \alpha_y, \alpha_z, \\ & \Sigma_x, \Sigma_y, \Sigma_z, \\ & \beta \Sigma_x, \beta \Sigma_y, \beta \Sigma_z, \\ & i\beta \alpha_x, i\beta \alpha_y, \text{ or } i\beta \alpha_z \end{aligned} \tag{6.2}$$

It is specified by a keyword indicating general form, and by specification of scalar operators Ω_j and factors a_i :

	Keyword	Operator form	Factors
1	DIAGONAL	$a_1 I_4 \Omega$	1
2	XALPHA	$a_1 \alpha_x \Omega$	1
3	YALPHA	$a_1 \alpha_y \Omega$	1
4	ZALPHA	$a_1 \alpha_z \Omega$	1
5	XAVECTOR	$a_1 \alpha_y \Omega_z - a_2 \alpha_z \Omega_y$	2
6	YAVECTOR	$a_1 \alpha_z \Omega_x - a_2 \alpha_x \Omega_z$	2
7	ZAVECTOR	$a_1 \alpha_x \Omega_y - a_2 \alpha_y \Omega_x$	2
8	ALPHADOT	$a_1 \alpha_x \Omega_x + a_2 \alpha_y \Omega_y + a_3 \alpha_z \Omega_z$	3
9	GAMMA5	$a_1 \gamma_5 \Omega$	1
10	XSIGMA	$a_1 \Sigma_x \Omega$	1
11	YSIGMA	$a_1 \Sigma_y \Omega$	1
12	ZSIGMA	$a_1 \Sigma_z \Omega$	1
13	XBETASIG	$a_1 \beta \Sigma_x \Omega$	1
14	YBETASIG	$a_1 \beta \Sigma_y \Omega$	1
15	ZBETASIG	$a_1 \beta \Sigma_z \Omega$	1
17	XiBETAAL	$a_1 i \beta \alpha_x \Omega$	1
18	YiBETAAL	$a_1 i \beta \alpha_y \Omega$	1
19	ZiBETAAL	$a_1 i \beta \alpha_z \Omega$	1

Operator forms 5–7 constitute the components of $(\boldsymbol{\alpha} \times \boldsymbol{\Omega})$, whereas operator form 8 is $(\boldsymbol{\alpha} \cdot \boldsymbol{\Omega})$. Operator 9 is the $\gamma_5 = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$ matrix. Factors are specified by

FACTORS define factors $\{a_i\}$

COMFACTO common factor for all components

The program will assume all operators to be Hermitian and will therefore insert an imaginary phase i if necessary (applies to antisymmetric scalar operators).

Operators are specified by the keyword `.OPERATOR` with the following arguments:

```
.OPERATOR
<operator name>
<operator type>
<labels for each component>
FACTORS
<factors for each component>
COMFACTOR
<common factor for all components>
```

Note that the arguments following the keyword `.OPERATOR` must start with a blank. The arguments are optional, except for the operator label. If no other arguments are given, the program assumes the operator to be a diagonal operator and expects the operator name to be the component label, for instance

```
.OPERATOR
OVERLAP
```

As an example the kinetic part of the Dirac Hamiltonian may be specified by:

```
.OPERATOR
'Kinetic energy'
ALPHAD
XDIPVEL
YDIPVEL
ZDIPVEL
COMFACTOR
-137.0359895
```

Another example is the finite field calculation with the \hat{z} dipole length operator added to the Hamiltonian $\mathbf{H} = \mathbf{H}_0 + 0.01 \cdot \hat{z}$

```
.OPERATOR
ZDIPLN
COMFACTOR
0.01
```

6.2 List of one-electron operators

- MOLFLD - Nuclear attraction integrals
- Symmetric
 - One component:
- MOLFIELD $\Omega_1 = \sum_I \hat{V}_{en}$
Point nucleus:
Finite nucleus:
- BETAMAT - overlap integrals; only SS-block
- Symmetric
 - One component:

	BETAMAT	$\Omega_1 = 1$; only SS-block
OVERLAP	- Overlap integrals	
	• Symmetric	
	• One component:	
	OVERLAP	$\Omega_1 = 1$
DIPLEN	- Dipole length integrals	
	• Symmetric	
	• Three components:	
	XDIPLEN	$\Omega_1 = x$
	YDIPLEN	$\Omega_2 = y$
	ZDIPLEN	$\Omega_3 = z$
DIPVEL	- Dipole velocity integrals	
	• Anti-symmetric	
	• Three components:	
	XDIPVEL	$\Omega_1 = \frac{\partial}{\partial x}$
	YDIPVEL	$\Omega_2 = \frac{\partial}{\partial y}$
	ZDIPVEL	$\Omega_3 = \frac{\partial}{\partial z}$
QUADRUP	- Quadrupole moments integrals	
	• Symmetric integrals	
	• Six components:	
	XXQUADRU	$\Omega_1 = \frac{\partial^2}{\partial x \partial x}$
	XYQUADRU	$\Omega_2 = \frac{\partial^2}{\partial x \partial y}$
	XZQUADRU	$\Omega_3 = \frac{\partial^2}{\partial x \partial z}$
	YYQUADRU	$\Omega_4 = \frac{\partial^2}{\partial y \partial y}$
	YZQUADRU	$\Omega_5 = \frac{\partial^2}{\partial y \partial z}$
	ZZQUADRU	$\Omega_6 = \frac{\partial^2}{\partial z \partial z}$
SPNORB	- Spatial spin-orbit integrals	
	• Anti-symmetric integrals	

	<ul style="list-style-type: none"> • Three components:
	$\text{X1SPNORB} \quad \Omega_1 = \sum_A \frac{l_x}{r_{iA}^3}$
	$\text{Y1SPNORB} \quad \Omega_2 = \sum_A \frac{l_y}{r_{iA}^3}$
	$\text{Z1SPNORB} \quad \Omega_3 = \sum_A \frac{l_z}{r_{iA}^3}$
SECMOM	- Second moments integrals
	<ul style="list-style-type: none"> • Symmetric integrals • Six components:
	$\text{XXSECMOM} \quad \Omega_1 = x^2$
	$\text{XYSECMOM} \quad \Omega_2 = xy$
	$\text{XZSECMOM} \quad \Omega_3 = xz$
	$\text{YYSECMOM} \quad \Omega_4 = y^2$
	$\text{YZSECMOM} \quad \Omega_5 = yz$
	$\text{ZZSECMOM} \quad \Omega_6 = z^2$
THETA	- Traceless theta quadrupole integrals
CARMOM	- Cartesian moments integrals
	<ul style="list-style-type: none"> • Symmetric integrals • $(l+1)(l+2)/2$ components ($l = i + j + k$)
	$\text{CMijjkk} \quad \Omega_l = x^i y^j z^k$
SPHMOM	- Spherical moments integrals (real combinations)
	<ul style="list-style-type: none"> • Symmetric integrals • $2l + 1$ components ($m = +0, -1, +1, \dots, +l$)
	$\text{SMl}\pm\text{mm} \quad \Omega_l = R_{l\pm m}$
SOLVENT	- Electronic solvent integrals
FERMI C	- One-electron Fermi contact integrals
PSO	- Paramagnetic spin-orbit integrals
SPIN-DI	- Spin-dipole integrals

DSO	- Diamagnetic spin-orbit integrals
SDFC	- Spin-dipole + Fermi contact integrals
HDO	- Half-derivative overlap integrals
S1MAG	- Second order contribution from overlap matrix to magnetic properties
S2MAG	- Contribution from overlap matrix to magnetic properties
ANGLON	- Angular momentum around the nuclei
ANGMOM	- Electronic angular momentum around the origin
LONMOM	- London orbital contribution to angular momentum
MAGMOM	- One-electron contributions to magnetic moment
KINENER	- Electronic kinetic energy
DSUSNOL	- Diamagnetic susceptibility without London contribution
DSUSLAN	- Angular London orbital contribution to diamagnetic susceptibility
DSUSLH	- Angular London orbital contribution to diamagnetic susceptibility
DIASUS	- Angular London orbital contribution to diamagnetic susceptibility
NUCSNLO	- Nuclear shielding integrals without London orbital contribution
NUCSLO	- London orbital contribution to nuclear shielding tensor integrals
NUCSHI	- Nuclear shielding tensor integrals
NEFIELD	- Electric field at the individual nuclei
ELFGRDC	- Electric field gradient at the individual nuclei, cartesian
ELFGRDS	- Electric field gradient at the individual nuclei, spherical
S1MAGL	- Bra-differentiation of overlap matrix with respect to magnetic field
S1MAGR	- Ket-differentiation of overlap matrix with respect to magnetic field
HDOBR	- Ket-differentiation of HDO-integrals with respect to magnetic field
NUCPOT	- Potential energy at the nuclei

HBDO	- Half B-differentiated overlap matrix
SQHDO	- Half-derivative overlap integrals not to be antisymmetrized
DSUSCGO	- Diamagnetic susceptibility with common gauge origin
NSTCGO	- Nuclear shielding integrals with common gauge origin
EXPIKR	- Cosine and sine integrals
MASSVEL	- Mass velocity integrals
DARWIN	- Darwin type integrals
CM1	- First order magnetic field derivatives of electric field
CM2	- Second order magnetic field derivatives of electric field
SQHDOR	- Half-derivative overlap integrals not to be anti-symmetrized
SQOVLAP	-

Appendix A

Recent Modifications

Nov 1 1998	Release of Dirac 3.1
Oct 6 2000	Release of Dirac 3.2
Sep 1 2004	Release of DIRAC 04.0

Index

**RESOLVE, 34
.2INDEX , 69
.4INDEX , 69
.4INDEX, 21

.A OPER, 61, 65
.ABUNDANCIES, 58
.ACMOUT, 23, 51
.ACTIVE, 68, 69
.ALLCMB, 61, 65
**ANA F, 22
**ANALYZE, 21, 22, 49
.ANALYZE, 21, 22
.ANGINT, 32
ANGLON , 108
.ANGMIN, 32
ANGMOM , 108
.AOLAB , 49, 50
.ATSIZE, 32, 33
.AUTOCC, 37

.B FREQ, 61, 65
.B OPER, 61, 65
BASDIR, 18
BETAMAT, 105
BETAMAT , 106

.C FREQ, 65
.C OPER, 65
CARMOM , 107
CITYPE, 81
.CLOSED SHELL ELECTRONS, 37
.CLOSED SHELL, 35

CM1 , 109
CM2 , 109
.CNVINT, 43, 62, 66
COMFACTO, 104
COMIN , 78
.COMPRES, 63
CONVERE , 77
CONVERR , 77
.CORE, 69
CPUMAX , 77
.CUBADJ, 53
.CVALUE, 24

.DAMPFC, 41
DARWIN , 109
DEBUG , 72
DEGEN, 79
DENSI, 81
*DENSIT, 52
.DENSIT, 49
DF1INT, 101
DFCMOS, 102
DFCOEF, 101, 102
DFCYCL, 101, 102
DFDIIS, 101, 102
DFEVEC, 101, 102
DFFCK1, 101
DFFCK2, 102
DFFOCK, 102
DFOVLP, 101
.DFPCMO, 23
*DFT, 32

- .DFT, 29, 34, 35
DFTMAT, 101
DFTWXX, 102
DFXXSA, 101
DFXXSB, 101
DFXXTA, 101
DFXXTB, 101, 102
.DHF , 34
*DHFCAL, 34, 35
DIASUS , 108
.DIISAO, 41
.DIISMO, 41
.DIISTH, 40
DIPLN , 106
.DIPLN, 65
.DIPOLE, 58
.DIPORG, 25
DIPVEL , 106
DIRAC.INP, 17
**DIRAC, 20, 21, 34, 68
.DIRECT, 23
DIRNOD, 17
DIRPAR, 17
DIRRCI, 76
**DIRRCI, 68
.DIRRCI, 34
DIRWRK, 17
DOCCSD , 72
DOCCSDT , 72
.DOCUBE, 53
DOEA , 74
DOEA2 , 74
DOENER , 71
DOFOPR , 71
DOFSPC , 71
DOIE , 74
DOIE2 , 74
.DOJACO, 23
DOMP2 , 72
DOMP2G , 73
DOSORT , 72
DSO , 108
.DSO, 59
DSUSCGO, 109
DSUSLAN, 108
DSUSLH , 108
DSUSNOL, 108
DUALFAMILY, 97
.EFG, 58
.EIGPRI, 44
ELFGRDC, 108
ELFGRDS, 108
.ERGCNV, 40
.ERRELS, 33
.EVCCNV, 40
*EXPECTATION VALUE, 60
EXPIKR , 109
EXPLICIT, 99
FACTORS , 104
FAMILY, 96
.FCKCNV, 40
FERMI C, 107
.FIRST ORDER HYPERPOLARIZABILITY, 59
.FIXDIF, 43
.FREEPJ, 31
.FROMVX, 33
.FROZEN, 45
FSSECT , 75
GASSHE, 82
GASSPC, 82
.GAUGEO, 63
**GENERAL, 23, 43, 46, 48, 62, 64, 65, 68,
69
GETDET , 77
GOSCIP, 79
**HAMILTONIAN, 20, 29, 34, 35
HBDO , 109

HDO , 108
HDOBR , 108
.HFXFAC, 30

.ICEDIF, 26
IGENEX , 77
.IJTSK, 46
INACTI, 81, 82
.INDSML, 50
INIWFC, 80
.INPTEST, 22
.INTCHK, 33
**INTEGRALS, 25–27, 58, 59
.INTFL2, 68
.INTFL4, 68, 69
.INTFLG, 30, 43, 46–48, 62, 64, 65, 68
INTGRL, 99
IPRNT, 79
IPRNT , 71
IREPNA , 76
ISTART , 76
.ITRINT, 43, 62, 66

KINENER, 108

.LABDEF, 50
LARGE, 96, 97
.LEVY-LEBLOND, 29
.LINDEP, 23
*LINEAR RESPONSE, 58, 60
LONMOM , 108
LUCITA, 80
.LUCITA, 34
.LVCORR, 29, 30
.LVNEW, 30

.MAGCOR, 25
MAGMOM , 108
MASSVEL, 109
MAXDIM , 73–75
MAXE3 , 76
MAXH1 , 76
MAXIT , 73–75
MAXITER , 77
.MAXITR, 40, 62, 66
.MAXPRI, 26
.MAXRED, 62, 66
.MDCINT , 70
.MESH , 56
MOLECULE.INP, 17, 27
MOLFIELD, 105
MOLFLD , 105
*MOLGRD, 22, 64
.MOLGRD, 59, 64
**MOLTRA, 21, 34, 68, 71
.MOLTRA, 68
.MP2 , 34
*MP2CAL, 34, 45
*MULPOP, 50
.MULPOP, 49
MULTIP, 81
.MVO, 34
*MVOCAL, 34, 48
.MVOVEC, 48
.MXDIIS, 41

NACTEL, 82
NACTH , 74
NACTP , 74
.NCUBE, 53
NEFIELD, 108
NELEC, 79
NELEC , 71, 76
NELEC_F1(2), 71
.NETPOP, 50
NFROZ , 72
*NMR, 63
.NMR, 59
.NO SKIP, 22
.NO2IND , 69
.NO4IND , 69

- .NO4INDEX, 21
- .NOBNCR , 39
- NOCCD , 73
- NOCCS , 73
- .NODAMP, 41
- .NODIIS, 41
- .NODSCF , 43
- .NODYNSEL, 41, 42
- .NOPREC, 63, 66
- .NOPRUN, 32
- .NOSCAT , 70
- .NOSMLV, 31
- .NQCC, 58
- NRAS1, 76
- NRAS2, 76
- NROOTS, 81
- NROOTS , 76
- NSEL(NROOTS) , 77
- NSTCGO , 109
- .NSTDIAMAGNETIC, 59
- NTOL , 72, 74, 75
- .NUCMOD, 25
- NUCPOT , 108
- NUCSHI , 108
- NUCSLO , 108
- NUCSNLO, 108
- .NUMGRA, 22, 64

- .OCCUPIED, 46
- .ONECAP, 31
- .ONECNV, 32
- *ONEINT, 26
- .ONESYS, 31
- .ONLY INTEGRALS, 22
- .ONLYSF, 63
- .ONLYSG, 63
- open shell DHF, 35
- .OPEN SHELL ELECTRONS, 37
- .OPEN SHELL, 35
- .OPERAT, 30, 60, 61, 70

- .OPERATOR, 104, 105
- *OPTIMIZE, 21, 22
- .OPTIMIZE, 21, 22
- .ORBANA, 60
- OVERLAP, 106
- OVERLAP , 106
- .OVLSEL, 42
- .OWNBAS, 45, 51

- pam, 14
- pamq, 15
- .PAR4BAS , 70
- .PCMOIN, 23
- .PCMOUT, 23
- .PHCOEF, 44
- .POLARIZABILITY, 58
- POPANA, 79
- .POSITRONS, 69
- .POST DHF REORDER MO'S, 42
- .PRICMP, 49
- .PRINT , 24–26, 43, 50, 52, 58, 60, 61, 66, 69, 70
- .PRINT, 27, 31, 32, 46–49, 53, 55, 64
- PRINTG, 81
- PRINTL, 81
- *PRIVEC, 49
- .PRIVEC, 49
- .PRJTHR, 45
- .PROJEC, 44, 49
- *PROJECTION, 51
- **PROPERTIES, 21, 22, 58
- .PROPERTIES, 21, 22
- .PROTHR, 52
- **PRP F, 22
- .PRPTRA , 69
- *PRPTRA, 70
- PSO , 107
- .PVC, 60

- *QUADRATIC RESPONSE, 64

- QUADRUP, 106
- .QUADRUPOLE, 58

- .RADINT, 32
- RAS1, 82
- RAS2, 82
- RAS3, 83
- .RCORBS , 69
- *READIN, 26
- .REAXVC, 62
- RELCCSD, 71
- .RELCCSD, 34
- .REORDER MO'S, 41, 42
- .RESFAC, 63, 66
- *RESOLVE, 47
- .RESOLVE, 34, 37, 79
- RESTART , 77
- *RHO1, 55
- .RHO1, 49
- RSTRCI, 81

- S, 88
- S1MAG , 108
- S1MAGL , 108
- S1MAGR , 108
- S2MAG , 108
- .SCATTER , 70
- .SCHEME , 48, 70
- .SCLMEM, 47
- .SCREEN , 69
- .SCREEN, 27, 39, 46-48
- SDFC , 108
- SECMOM , 107
- SELECT , 77
- .SELECT, 25, 58, 59
- SELPOP, 79
- .SHIELDING, 59
- .SKIPEE, 62, 65
- .SKIPEP, 62, 66
- .SMLV1C, 31

- .SOFOCK, 26
- SOLVENT, 107
- SPHMOM , 107
- .SPHTRA, 24
- SPIN-DI, 107
- .SPIN-SPIN COUPLING, 59
- .SPINFREE, 29, 30
- SPNORB , 106
- SQHDO , 109
- SQHDOR , 109
- SQOVLAP, 109
- .STERNH, 63
- SYMMET, 81
- SZCALC, 81

- THETA , 107
- THRESH, 79
- .THRESH, 62, 66
- .THRFACT, 27
- .THROUT , 69
- .TIME, 27
- TIMING , 72
- TITLE, 81
- title line, 21
- .TITLE, 21
- .TRIAB, 61
- .TRICK, 22, 64
- TRIFCK, 102
- .TRIFCK , 39
- .TRIVFC , 39
- .TWO-PHOTON, 59
- .TWOGRD, 22
- *TWOINT, 26, 39

- .UNCONTRACT , 26
- .URKBAL, 31
- USEOE , 73

- .VECMVO, 48
- .VECPOP, 50
- .VECPRI, 49

.VECPRJ, 51
.VECREF, 51
.VERDET, 59
.VEXTPJ, 31
.VIRTHR, 46
.VIRTUAL, 46

.WAVE F, 34
**WAVE FUNCTION, 21
.WAVE FUNCTION, 21
**WAVE FUNCTIONS, 34
.WGPOP, 52

X1SPNORB , 107
XDIPLN , 106
XDIPVEL , 106
.XLRNRM, 63
.XQRNRM, 65
XXQUADRU , 106
XXSECMOM , 107
XYQUADRU , 106
XYSECMOM , 107
XZQUADRU , 106
XZSECMOM , 107

Y1SPNORB , 107
YDIPLN , 106
YDIPVEL , 106
YYQUADRU , 106
YYSECMOM , 107
YZQUADRU , 106
YZSECMOM , 107

Z1SPNORB , 107
ZDIPLN , 106
ZDIPVEL , 106
.ZORA, 30
ZZQUADRU , 106
ZZSECMOM , 107

Bibliography

- [1] L. Visscher and K. G. Dyall. Dirac-fock atomic electronic structure calculations using different nuclear charge distributions. *At. Data Nucl. Data Tabl.*, 67:207, 1997.
- [2] T. Saue, K. Fægri, T. Helgaker, and O. Gropen. Principles of direct 4-component relativistic scf: Application to cesium auride. *Mol. Phys.*, 91:937, 1997.
- [3] L. Visscher and T. Saue. Approximate relativistic electronic structure methods based on the quaternion modified dirac equation. *J. Chem. Phys.*, 113:3996, 2000.
- [4] J. M. Lévy-Leblond. Nonrelativistic particles and wave equations. *Commun. Math. Phys.*, 6:286, 1967.
- [5] K. G. Dyall. An exact separation of the spin-free and spin-dependent terms of the dirac-coulomb-breit hamiltonian. *J. Chem. Phys.*, 100:2118, 1994.
- [6] L. Visscher. Approximate molecular dirac-coulomb calculations using a simple coulombic correction. *Theor. Chem. Acc.*, 96:68–70, 1997.
- [7] P. A. M. Dirac. Note on exchange phenomena in the thomas atom. *Proc. Roy. Soc. London*, 26:376, 1930.
- [8] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38:3098, 1988.
- [9] J. P. Perdew and Y. Wang. Accurate and simple density functional for the electronic exchange energy: Generalized gradient approximation. *Phys. Rev. B*, 33:8800, 1986.
- [10] S. J. Vosko, L. Wilk, and M. Nusair. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: A critical analysis. *Can. J. Phys.*, 58:1200, 1980.
- [11] C. Lee, W. Yang, and R. G. Parr. Development of the colle-salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, 37:758, 1988.

- [12] J. P. Perdew. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B*, 33:8822, 1986.
- [13] E. van Lenthe, E. J. Baerends, and J. G. Snijders. Relativistic total energies using regular approximations. *J. Chem. Phys.*, 101:9783, 1994.
- [14] E. van Lenthe, J. G. Snijders, and E. J. Baerends. The zero-order regular approximation for relativistic effects: The effect of spin-orbit coupling in closed shell molecules. *J. Chem. Phys.*, 105:6505, 1996.
- [15] T. Saue and T. Helgaker. Four-component relativistic kohn–sham theory. *J. Comput. Chem.*, 23:814, 2002.
- [16] O. Fossgaard, O. Gropen, M. Corral Valero, and T. Saue. On the performance of four-component relativistic density functional theory: Spectroscopic constants and dipole moments of the diatomics hx and xy (x,y=f, cl, br and i). *J. Chem. Phys.*, 118:10418, 2003.
- [17] M. Mayer, O. D. Häberlen, and N. Rösch. *Phys. Rev. A*, 54:4775, 1996.
- [18] S. Varga, E. Engel, W.-D. Sepp, and B. Fricke. *Phys. Rev. A*, 59:4288, 1999.
- [19] S. Varga, B. Fricke, H. Nakaamatsu, T. Mukoyama, J. Anton, D. Geschke, A. Heitmann, and E. Engel et T. Bastug. *J. Chem. Phys.*, 112:3499, 2000.
- [20] A. D. Becke. A multicenter numerical integration scheme for polyatomic molecules. *J. Chem. Phys.*, 88:2547, 1988.
- [21] R. Lindh, P. A. Malmqvist, and L. Galgardi. Molecular integrals by numerical quadrature. i. radial integration. *Theor. Chem. Acc.*, 106:178, 2001.
- [22] T. Saue and H. J. Aa Jensen. Quaternion symmetry in relativistic molecular calculations: I. the dirac-fock method. *J. Chem. Phys.*, 111:6211, 1999.
- [23] T. Saue and H. J. Aa. Jensen. Quaternion symmetry of the dirac equation. In M. De-francheschi and C. Le Bris, editors, *Mathematical Methods for Ab Initio Quantum Chemistry*, volume 74 of *Lecture Notes in Chemistry*, page 227. Springer, Berlin, 2000.
- [24] J. Thyssen and H. J. Aa. Jensen. Average-of-configurations SCF manuscript, 1998.
- [25] P. Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73:393, 1980.
- [26] P. Pulay. Improved scf acceleration. *J. Comput. Chem.*, 3:556, 1982.

- [27] T. Hamilton and P. Pulay. Direct inversion in the iterative subspace(diis) optimization of open-shell, excited-state, and small multiconfiguration scf wave functions. *J. Chem. Phys.*, 84:5728, 1986.
- [28] W. Kutzelnigg. Chemical bonding in the higher main group elements. *Angew. Chem. Int. Ed. Engl.*, 23:272, 1984.
- [29] C. W. Bauschlicher. *J. Chem. Phys.*, 72:880, 1980.
- [30] K. Faegri and T. Saue. Diatomic molecules between very heavy elements of group 13 and group 17: A study of relativistic effects on bonding. *J. Chem. Phys.*, 115:2456, 2001.
- [31] J. K. Laerdahl and P. Schwerdtfeger. Fully relativistic ab initio calculations of the energies of chiral molecules including parity-violating weak interactions. *Phys. Rev. A*, 60:4439, 1999.
- [32] T. Saue and H. J. Aa. Jensen. Linear response at the 4-component relativistic level: Application to the frequency-dependent dipole polarizabilities of the coinage metal dimers. *J. Chem. Phys.*, 118:522, 2003.
- [33] T. Saue. Post dirac-hartree-fock methods - properties. In P. Schwerdtfeger, editor, *Relativistic Electronic Structure Theory - Part 1 : Fundamentals*. Elsevier, Amsterdam, 2002.
- [34] L. Visscher. The dirac equation in quantum chemistry : strategies to overcome the current computational problems. *J. Comput. Chem.*, 23:759, 2002.
- [35] L. Visscher, T. J. Lee, and K. G. Dyall. Formulation and implementation of a relativistic unrestricted coupled cluster method including noniterative connected triples. *J. Chem. Phys.*, 105:8769, 1996.
- [36] M. Pernpointner and L. Visscher. Parallelization of four-component calculations. ii. symmetry- driven parallelization of the 4-spinor ccsd algorithm. *J. Comput. Chem.*, 24:754, 2003.
- [37] L. Visscher, E. Eliav, , and U. Kaldor. Formulation and implementation of the relativistic fock-space coupled cluster method for molecules. *J. Chem. Phys.*, 115:9720, 2001.
- [38] L. Visscher, O. Visser, H. Aerts, H. Merenga, and W. C Nieuwpoort. Relativistic quantum chemistry: the molfdi program package. *Comput. Phys. Comm.*, 81:120, 1994.

- [39] T. Fleig and L. Visscher. Large-Scale Electron Correlation Calculations in the Framework of the Spin-Free Dirac Formalism. The Au₂ Molecule Revisited. *Chem. Phys.*, 2004.
- [40] K. G. Dyall, I. P. Grant, C. T. Johnson, F. A. Parpia, and E. P. Plummer. Grasp: A general-purpose relativistic atomic structure program. *Comput. Phys. Comm.*, 55:425, 1989.