LSDALTON 2011 Program Manual

S. Coriani, T. Helgaker, S. Høst,
B. Jansík, P. Jørgensen, J. Kauczor,
T. Kjærgaard, K. Kristensen, J. Olsen,
S. Reine, P. Sałek, A. Thorvaldsen,
L. Thøgersen, V. Rybkin,
V. Bakken, M. Watson,
and A. Krapp

Contents

Preface

This is the manual for the LSDALTON quantum chemistry program — Release 2011 — for computing Hartree-Fock and DFT wave functions, energies, and molecular properties. The focus of LSDALTON is linear scaling in all parts of the code, which makes it suitable for calculations on large molecular systems. For the actual documentation of the code, see `http://www.dirac.chem.sdu.dk/tracDalton`.

We emphasize the conditions under which the program is distributed. It is furnished for your own use, and you may not redistribute it further, either in whole or in part. Even though LSDALTON is completely separate from the original DALTON program, the program packages are distributed together. Thus, anyone interested in obtaining LSDALTON should check out the DALTON homepage at `http://www.kjemi.uio.no/software/dalton/dalton.html`.

Any use of the program that results in published material should cite two papers which are expected to be out in the spring of 2011. At that point, the references will be put both here and in the DALTON.OUT output file from an LSDALTON calculation.

The program represents experimental code that is under constant development. No guarantees of any kind are provided, and the authors accept no responsibility for the performance of the code or for the correctness of the results.

Part I

DALTON Installation Guide

Chapter 1

Installation

1.1 Hardware/software supported

LSDALTON can be run on a variety of systems running the UNIX operating system. The current release of the program has been tested on IBM-AIX (using xlf90/xlc), Linux (using ifort/icc and gfortran/gcc), and to some extent Mac/Darwin (using gfortran/gcc), but is expected to work on other architectures as well. The program is written mainly in Fortran 90. The exception is the DFT grid generation and functional evaluation in the exchange-correlation part, which is written in C.

1.2 Source files

LSDALTON is distributed as a `tar` file obtainable from the DALTON homepage (`http://www.kjemi.uio.no/software/dalton/dalton.html`) provided the license agreement for the program has first been completed and returned to the authors. If you have accessed this documentation off the `tar` file you will already know how to extract the required directory structure, but for completeness, assuming the `tar` file is called `dalton.tar.gz`, the commands

```
gunzip dalton.tar.gz
tar xf dalton.tar
```

will produce the following subdirectory structure in the current directory:

```
dalton/basis         dalton/LSint      dalton/mm
dalton/dft           dalton/linears    dalton/pdpack
dalton/doc_LSDALTON  dalton/lsutil     dalton/test
dalton/geomopt
```

Most of the subdirectories contain source code for the different sections constituting the program (`dft`, `linears`, `LSint`, `lsutil`, `geomopt` and `mm`). Furthermore, there's a directory containing various public domain routines (`pdpack`) to be used if no mathematical libraries are available, a directory containing all the basis sets supplied with this distribution (`basis`), a fairly large set of test jobs including reference output files (`test`), and finally this documentation (`doc_LSDALTON`).

In addition to the directories, the main dalton directory will contain several files including a shell script (`configure`) which will build a suitable `Makefile.config` for use when installing the program. The configure script will also create a script `dalton` for running the test jobs.

1.3 Installing the program using the Makefile

The program should be easily installed through the use of the supplied `configure` script. Based on the automatic detection of the architecture, the script will try to build a suitable `Makefile.config` on the basis of what kind of mathematical libraries are found, and user input. Thus, to execute the script, type

```
> ./configure
```

The script will now try to detect the architecture, which usually works fine for most common platforms. However, if for some reason this is not possible, you may choose one of the supported architectures from a list (currently comprising only three architectures):

aix	linux	darwin

Although this script in most cases is capable of making a correct `Makefile.config`, we always recommend users to check the created `Makefile.config` against local system set-up. In particular, check that `Makefile.config` contains the path(s) to the proper mathematical libraries. The program runs many times slower if not linked to any mathematical libraries (i.e. using LSDALTON's own "hand coded" matrix multiplications etc.)

During the execution of the `configure` script, you will be asked to decide which compile you would like to use.

The configure script searches for compilers in a orded fashion, and in each instance you can chose to accept the compiler.

For linux

Fortran77 (ifort,ifc,efc,pgf77,gfortran)

Fortran90 (ifort,ifc,efc,pgf90,gfortran)

C (icc,ecc,pgcc,gcc)

For darwin

Fortran77 (xlf,g77,f77,gfortran)

Fortran90 (gfortran,xlf90,xlf)

C (xlc,gcc)

For AIX

Fortran77 (xlf,f77,f90)

Fortran90 (xlf90,f90)

C (xlc,cc)

When a suitable compiler have been found, you will be asked a few questions:

1. **Do you want to install the program with OpenMP parallelization?**

   Specifies wether or not you would like to compile the code using openMP (utilizing several cores (or processors) on a single node. Note that MPI parallelization is not (yet) supported by LSDALTON.

2. **Use current directory as installation directory for binaries and scripts?**

   Denotes the directory where the executable and the run script will be moved to.

3. **Use $installdir/basis as default basis set directory?**

   This defines the directory where the program will look for the basis sets supplied with the distribution, and this need to be changed according to the local directory structure. We recommend that the basis sets in this directory are *not* changed. Changes to the basis set should be done in a separate directory, and you may then supply this basis set directory to the program at execution time.

4. **Default scratch space**

   Determines the default head scratch directory where temporary files will be placed. This value will be put in the `dalton` run script. However, note that jobs will be run in a subdirectory of this head scratch-directory, according to the name of the job files. If `/work` or `/scratch` is defined in the local directory structure, the script will normally suggest `/work/$USER` or `/scratch/$USER` as default head scratch space.

Compiler options will be supplied in `Makefile.config`. When `Makefile.config` has been properly created and checked to agree with local system set-up, all that is needed in order to to build an executable version of the code is to type (in the same directory as the `Makefile.config` file):

```
> make
```

1.4 The Makefile.config

In this section we give a brief introduction to the Makefile.config, which contains compiler instructions for the compilation of the program. The introduction is directed to new users with a limited understanding of Makefiles and we only discuss subjects that may be relevant. Note that the configure script will in most cases provided a Makefile.config which works and no modifications are required from the user. Depending on the version of math libraries, the configure script may not be able to find these and it may be required to manually add paths to mathematical libraries.

1.4.1 Intel compiler

The first lines of a Makefile.config using the intel ifort/icc compilers may look like this

```
1   ARCH       = linux
```

```
2   FMMDIR       = mm
3   #
4   #
5   CPPFLAGS      = -DSYS_LINUX -D_FILE_OFFSET_BITS=64
    -D'INSTALL_BASDIR="/home/user/lsdalton/basis/"' -DVAR_LINSCA
6   F77            = ifort
7   F90            = ifort
8   FLNK           = ifort
9   CC             = icc
10  RM             = rm -f
11  FFLAGS         = -O3 -xW -ip -w
12  F90OPTFLAGS    = -O3 -xW -ip -w -fpp1 -openmp
13  SAFEFFLAGS     = -O2 -w
14  CFLAGS         = -O3 -xW -ip -restrict -DRESTRICT=restrict -DUSE_UNDERSCORES
15  INCLUDES       =
16  LIBS           = -lguide -lpthread
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o gp_dlapack.o gp_zlapack.o gp_dblas3.o
    gp_dblas2.o gp_dblas1.o gp_zblas.o
```

We now describe this Makefile.config file line by line, before we discuss a number of issues.

- Line 1: ARCH is set to the architecture. (for an aix architecture ARCH=rs6000, for a Mac computer ARCH=darwin, these will be discussed shortly)
- Line 2: The directory of the Fast multipole moment code
- Line 5: CPPFLAGS is a list of precompiler flags which is required for the program to compile correctly

  -DSYS_LINUX tells the dalton program that this is a linux architecture

  -DINSTALL_BASDIR The basisset installation directory

  -DVAR_LINSCA Is a mandatory keyword!

  Usually this line works out of the box
- Line 6: specifies the compiler used to compile fortran 77 files
- Line 7: specifies the compiler used to compile fortran 90 files
- Line 8: specifies the compiler used for linking

- Line 9: specifies the compiler used to compile c files
- Line 10: specifies the command to be used when cleaning
- Line 11-13: specifies compiler flags that should be used to compile fortran files. Naturally these flags depend on the compiler. Here we use a intel fortran compiler (ifort) 64 bit version 11.1.056

  -O3 Specifies the code optimization for applications. Possible options are O0, O1, O2, or O3, with O2 as default

  -xW Tells the compiler to generate optimized code specialized for the processor that executes your program. This option is still supported in version 11.1.05, but are planned to be unsupported in future releases

  -ip Additional interprocedural optimizations for single-file compilation are enabled.

  -w Disables all warning messages (same as -nowarn)

  -fpp1 Runs the Fortran preprocessor on source files before compilation. Syntax is fpp[n], where n can be 0, 1, 2, or 3, where 1, 2, or 3 tells the compiler to run the preprocessor

  -openmp Enables the parallelizer to generate multi-threaded code based on the OpenMP directives. This option sets option automatic, which causes all local, non-saved variables to be allocated to the run-time stack, this may result in a lack of stack-memory as discussed below.

- Line 14: specifies compiler flags that should be used to compile C files. Naturally these flags depend on the compiler. Here we use a intel fortran compiler (icc) 64 bit version 11.1.056. Some of the compiler options have already been explained under the ifort options and they will not be explained again.

  -restrict Pointer disambiguation is enabled with the restrict qualifier.

  -DRESTRICT=restrict Precompiler flag used in DFT code

  -DUSE_UNDERSCORES required to define the communication between fortran and C files.

- Line 15: Empty by default. Used if for some reason, any additional directories should be included.
- Line 16: Libraries you wish to include. -lguide and -lpthread are required for openmp parallelization. See Section 1.4.2 for more details on how to link to mkl and other libraries

- Line 17: the install directory
- Line 18: LSDALTON provides its own blas and lapack libraries, located in the pdpack directory. These should only be used if no mathematical libraries are available, since they slow down the code significantly.

  In case you link to an external library, like the intel mkl library, you only need to include "linpack.o eispack.o".

  In case you do not link to an external library, you need to include "linpack.o eispack.o. gp_dlapack.o gp_zlapack.o gp_dblas3.o gp_dblas2.o gp_dblas1.o gp_zblas.o."

OpenMP

Note that when compiling using the -openmp option it may be required to increase the stack memory. On a linux machine the size of the stack-memory can be viewed using the command

```
> ulimit -a
```

and changed by

```
> ulimit -s newstacksize
```

The newstacksize should be a large number or unlimited.

Math Library: MKL

Linking to a math library is crucial for optimal performance. The linking depends on the library available. We encourage to first compile with the default Makefile.config composed by the configure script. This ensures that your chosen compiler works with dalton, and then you can try to link to a math library. There is a number of different ways that can be used to link the mkl library to dalton. One brute force way is the following:

```
16  LIBS           =
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_solver_lp64_sequential.a
-Wl,--start-group /opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_intel_lp64.a
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_sequential.a
```

```
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_core.a -Wl,--end-group
-lguide -lpthread
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

for the sequential linking (no OpenMP), assuming that the intel compiler is located at /opt/intel/Compiler/11.1/056 and that the mkl library is located at /opt/intel/Compiler/11.1/056. The OpenMP version of the library can be used in the following way.

```
16  LIBS           =
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_solver_lp64_sequential.a
-Wl,--start-group /opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_intel_lp64.a
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_intel_thread.a
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_core.a -Wl,--end-group
-lguide -lpthread
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

A somewhat more elegant way is to add the MKL directory to the INCLUDE line

```
15  INCLUDES       = -I/home/user/lsdalton/include
16  LIBS           = -I/opt/intel/Compiler/11.1/056/mkl/include -lmkl_intel_lp64
-lmkl_sequential -lmkl_core
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

or a threaded version

```
15  INCLUDES       = -I/home/user/lsdalton/include
16  LIBS           = -I/opt/intel/Compiler/11.1/056/mkl/include -lmkl_intel_lp64
-lmkl_intel_thread -lmkl_core -lguide -lpthread
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

1.4.2 Gfortran/gcc compiler

The first lines of a Makefile.config using the gfortran/gcc (version 4.5.1) on a linux architecture may look like this

```
ARCH       = linux
FMMDIR      = mm
#
#
CPPFLAGS      = -DSYS_LINUX -D_FILE_OFFSET_BITS=64
-D'INSTALL_BASDIR="/home/user/lsdalton/basis/"' -DVAR_LINSCA
F77           = gfortran
F90           = gfortran
FLNK          = gfortran
CC            = gcc
RM            = rm -f
FFLAGS        = -march=x86-64 -O3 -ffast-math -funroll-loops
-ftree-vectorize -ffloat-store
F90OPTFLAGS   = -march=x86-64 -O3 -ffast-math -funroll-loops
-ftree-vectorize -ffloat-store -I. -x f95-cpp-input -ffloat-store -fopenmp
SAFEFFLAGS    = -march=x86-64 -O3 -ffast-math -funroll-loops
-ftree-vectorize -ffloat-store
CFLAGS        = -march=x86-64 -O3 -ffast-math -funroll-loops
-ftree-vectorize -std=c99 -DRESTRICT=restrict -DFUNDERSCORE=1 -ffloat-store
-DUSE_UNDERSCORES
INCLUDES      = -I/home/user/lsdalton/include
LIBS          = -lgomp
INSTALLDIR    = /home/user/lsdalton
PDPACK_EXTRAS  = linpack.o eispack.o gp_dlapack.o gp_zlapack.o gp_dblas3.o
gp_dblas2.o gp_dblas1.o gp_zblas.o
```

On a MAC computer a Makefile.config using gfortran/gcc could look like this

```
ARCH       = darwin
FMMDIR     = mm
#
#
CPPFLAGS      = -DSYS_LINUX -D_FILE_OFFSET_BITS=64
-DVAR_SPLITFILES -DVAR_G77 -DGFORTRAN -DVAR_LINSCA
-D'INSTALL_BASDIR="/home/user/lsdalton/basis/"'
F77           = gfortran
F90           = gfortran
FLNK          = gfortran
```

```
9   CC             = gcc
10  RM             = rm -f
11  FFLAGS         = -O3 -ffast-math -fexpensive-optimizations -funroll-loops
    -ftree-vectorize
12  F90OPTFLAGS    = -O3 -ffast-math -fexpensive-optimizations -funroll-loops
    -ftree-vectorize -I. -x f95-cpp-input
13  SAFEFFLAGS     = -O2 -ffast-math -fexpensive-optimizations -funroll-loops
    -ftree-vectorize
14  CFLAGS         = -O3 -ffast-math -fexpensive-optimizations -funroll-loops
    -ftree-vectorize -std=c99 -DRESTRICT=restrict -DFUNDERSCORE=1 -DUSE_UNDERSCORES
15  INCLUDES       = -I/home/user/lsdalton/include
16  LIBS           = -lpthread -lm -lgfortranbegin -lgfortran
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o gp_dlapack.o gp_zlapack.o gp_dblas3.o
    gp_dblas2.o gp_dblas1.o gp_zblas.o
```

We now describe these Makefile.config files in comparison to section 1.4.1.

Line 1 to 5 is unchanged for the linux architecture, but on the MAC machine -DGFORTRAN must be included in the CPPFLAGS.

Line 6-9 specifies the use of the gfortran/gcc compilers.

Line 11-13 specify compiler flags and are naturally different for different compilers

-march=x86-64 This specifies the name of the target architecture. GCC uses this name to determine what kind of instructions it can emit when generating assembly code. In this case it is a 64 bit machine. The use of this compiler flag is not required, but recommended.

-O3 Specifies the code optimization for applications. Possible options are O0, O1, O2, or O3, with O2 as default

-ffast-math A Compiler option to optimize floating-point arithmetic

-funroll-loops Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop

-ftree-vectorize Perform loop vectorization on trees. This flag is enabled by default at -O3

fexpensive-optimizations Perform a number of minor optimizations that are relatively expensive.

-ffloat-store This may be required depending on the gfortran version, but usually not. it enforces IEEE compliance

-fopenmp Enables the parallelizer to generate multi-threaded code based on the OpenMP directives.

-std=c99 Determines the language standard, in this case C 99 standard.

Line 16 are libraries you wish to include. -lgomp are required for openmp parallelization.

Math Library: Linux MKL

Linking to a math library is crucial for optimal performance. The linking depends on the library available. We encourage to first compile with the default Makefile.config composed by the configure script. This ensures that your chosen compiler works with dalton, and then you can try to link to a math library. Dalton provide its own blas and lapack libraries, which is placed in the pdpack directory. However it is encouraged to link to an external library, like the intel mkl library, then you only need to include linpack.o eispack.o. In the case that you do not link to an external library, you need to include all. linpack.o eispack.o. gp_dlapack.o gp_zlapack.o gp_dblas3.o gp_dblas2.o gp_dblas1.o gp_zblas.o.

Note that linking to mkl library using OpenMP requires to use -liomp5 in stead of -lgomp

```
15  INCLUDES       = -I/home/user/lsdalton/include
-I/opt/intel/Compiler/11.1/056/mkl/include
16  LIBS           = -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
-lguide -lpthread -liomp5
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

or

```
15  INCLUDES       =
16  LIBS           =
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_solver_lp64_sequential.a
-Wl,--start-group /opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_intel_lp64.a
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_intel_thread.a
/opt/intel/Compiler/11.1/056/mkl/lib/em64t/libmkl_core.a -Wl,--end-group
```

```
-lguide -lpthread -liomp5
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o
```

1.4.3 XLF/XLC

The Xlf/xlc compiler is usually used on AIX architectures and it may look like this for a serial (No OpenMP version)

```
1   ARCH       = rs6000
2   FMMDIR      = mm
3   #
4   #
5   CPPFLAGS      = -WF,-DSYS_AIX,-D'INSTALL_BASDIR="/home/user/lsdalton/basis/"' -DVAR_LINSCA
6   F77            = xlf
7   F90            = xlf90
8   FLNK           = xlf
9   CC             = xlc
10  RM             = rm -f
11  FFLAGS         = -O3 -qstrict -qarch=auto -qtune=auto -q64 -qextname
12  F90OPTFLAGS    = -O3 -qstrict -qarch=auto -qtune=auto -q64 -qextname
    -qsuffix=f=f90:cpp=f90 -qlanglvl=90std -qinit=f90ptr
13  SAFEFFLAGS     = -O2 -qstrict -qarch=auto -qtune=auto -q64 -qextname
14  CFLAGS         = -O3 -I/home/user/lsdalton/include -DSYS_AIX -D_LARGE_FILES
    -qlanglvl=stdc99 -DRESTRICT=restrict -qarch=auto -qtune=auto -q64
    -DUSE_UNDERSCORES
15  INCLUDES       = -I/home/user/lsdalton/include
16  LIBS           =
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o gp_dlapack.o gp_zlapack.o
    gp_dblas3.o gp_dblas2.o gp_dblas1.o gp_zblas.o
19  AR             = ar
20  ARFLAGS        = -X64 rvs
```

We now describe this Makefile.config in comparison to section 1.4.1 Line 1 to 4 is unchanged, but in line 5 the -DSYS_AIX is needed and a different syntax is used.

Line 6-9 specifies the use of the xlf/xlc compilers. Line 11-14 specifies the xlf/xlc recommended compiler option

-march=x86-64 This specifies the name of the target architecture. GCC uses this name to determine what kind of instructions it can emit when generating assembly code. In this case it is a 64 bit machine

-O3 Specifies the code optimization for applications. Possible options are O0, O1, O2, O3, O4 or O5, with O2 as default

-qstrict Ensures that optimizations done by default at optimization levels -O3 and higher, and, optionally at -O2, do not alter the semantics of a program

-qarch=auto specifies the processor architecture for which the code (instructions) should be generated. In this case the specific architecture the compiling machine is automatically detected.

-qtune=auto Optimizations are tuned for the platform on which the application is compiled

-q64 Indicates 64-bit compilation bit mode and, together with the -qarch option, determines the target machines on which the 64-bit executable will run

-q32 Enables 32-bit compilation mode (or, more briefly, 32-bit mode) support in a 64-bit environment

-qextname Adds an underscore to the names of all global entities. Used on the OpenMP directives.

-qlanglvl=90std Determines whether source code and compiler options should be checked for conformance to a specific language standard. In this case fortran 90 standard and C 99 standard.

-qinit=f90ptr Makes the initial association status of pointers disassociated.

In line 20 it may be necessary to manual add -X64 depending on the architecture (32 or 64 bit).

In case of OpenMP, a number of changes must be made. In line 7, the xlf90_r compiler is specified. This is the fortran compiler which is related to OpenMP. In addition the -qsmp=omp compiler flags must be used.

```
1   ARCH       = rs6000
2   FMMDIR      = mm
3   #
4   #
5   CPPFLAGS      = -WF,-DSYS_AIX,-D'INSTALL_BASDIR="/home/user/lsdalton/basis/"',-DVAR_LINSCA
6   F77            = xlf
7   F90            = xlf90_r
8   FLNK           = xlf
9   CC             = xlc
10  RM             = rm -f
11  FFLAGS         = -O3 -qstrict -qarch=auto -qtune=auto -q64 -qextname
12  F90OPTFLAGS    = -O3 -qstrict -qarch=auto -qtune=auto -q64 -qextname
    -qsuffix=f=f90:cpp=f90 -qlanglvl=90std -qinit=f90ptr -qsmp=omp
13  SAFEFFLAGS     = -O2 -qstrict -qarch=auto -qtune=auto -q64 -qextname
14  CFLAGS         = -O3 -I/home/user/lsdalton/include -DSYS_AIX -D_LARGE_FILES
    -qlanglvl=stdc99 -DRESTRICT=restrict -qarch=auto -qtune=auto -q64
    -DUSE_UNDERSCORES
15  INCLUDES       = -I/home/user/lsdalton/include
16  LIBS           =
17  INSTALLDIR     = /home/user/lsdalton
18  PDPACK_EXTRAS  = linpack.o eispack.o gp_dlapack.o gp_zlapack.o
    gp_dblas3.o gp_dblas2.o gp_dblas1.o gp_zblas.o
19  AR             = ar
20  ARFLAGS        = -X64 rvs
```

Math Library: MASS ESSL

For AIX architectures we recommend linking to the IBM Mathematical Acceleration Subsystem (MASS) library, which are shipped with the XL C, XL C/C++, and XL Fortran compiler products. This is done by adding the -lmass -lmassv -L/usr/local/mass flags to line 16. We also recommend the use of Engineering and Scientific Subroutine Library. The ESSL provides a set of highly optimized mathematical subroutines especially tuned for the IBM RISC System/6000 and for IBM Power 4. The library can be used with Fortran, C, and C++ programs. Since blas libraries are found through MASS and ESSL, only the lapack library routines from dalton are needed ("gp_zlapack.o gp_dlapack.o" in PDPACK_EXTRAS, in addition to "linpack.o eispack.o").

```
16  LIBS           = -lessl -lmass -lmassv -L/usr/local/mass
```

```
INSTALLDIR     = /home/user/lsdalton
PDPACK_EXTRAS  = linpack.o eispack.o gp_zlapack.o gp_dlapack.o
```

In the case of OpenMP special OpenMP libraries must be used

```
LIBS           = -Llib -L/usr/lib -lxlsmp  -lmass -lmassv -L/usr/local/mass
 -lpthread -lC -lesslsmp
INSTALLDIR     = /home/user/lsdalton
PDPACK_EXTRAS  = linpack.o eispack.o gp_zlapack.o gp_dlapack.o
```

1.4.4 Using 64-bit integers

For large scale calculations, it may be necessary to compile the program using 64-bit integers instead of the default 32-bit integers. To do this, you need to include ”-DVAR_INT64 -DVAR_64BITS” in the CPPFLAGS (in addition to what is already there). Furthermore, the FFLAGS, F90OPTFLAGS, SAFEFFLAGS, and CFLAGS must be changed depending on you compiler. For ifort/icc, you need to add ”-i8”, and for XLF/XLC, you need to add ”-qintsize=8”. The use of 64-bit integers has not been tested for gfortran. Note that you also have to change the mathematical libraries in LIBS to their 64-bit counterparts. This is not always a simple task, and you may need to ask your system administrator.

1.4.5 Using Compressed-Sparse Row matrices

When using Compressed-Sparse Row (CSR) matrices, only non-zero elements in matrices are stored. When matrices are sparse enough, linear scaling in both memory and cpu time is obtained. However, for small or non-sparse systems, there is a considerable overhead in cpu time compared to using standard dense matrices, so don't use it unless you need it. You enable CSR by putting ”-DVAR_MKL” in your CPPFLAGS (in addition to what is already there. You also need to link to MKL's CSR library (what we have in LSDALTON is an interface to MKL CSR). Furthermore, to run a calculation using CSR matrices, you always need .CSR under *DENSOPT in the input file DALTON.INP (described in detail in chapters 3 and 4).

1.5 Running the DALTON test suite

To check that LSDALTON has been successfully installed, a fairly elaborate automatic test suite is provided in the distribution. A test script and all the test jobs and reference output files can be found in the `dalton/test` directory. It is highly recommended that all these tests be run once the program has been compiled. Depending on your hardware, this usually takes 1/2—2 hours.

The tests can be run one by one or in groups, by using the test script `TEST`. Try `TEST -h` to see the different options this script takes. To run the complete test suite, go to the `dalton/test` directory and type:

```
> ./TEST all
```

You can follow the progress of the tests directly, but all messages are also printed to a log (`TESTLOG` by default). After all the tests have completed you should hopefully be presented with the message “ALL TESTS ENDED PROPERLY!”. If not, the complete list of failing test-cases will be printed. Please consult the file `KNOWN_PROBLEMS` too see if these tests have documented problems on your particular platform.

Any tests that fail will leave behind the `.mol` and `.dal` input-files (these are described in more detail in chapter 3), and the output file from the test calculation which will have the extension `.log` For all successfull tests these files, as well as some other auxiliary files, will be deleted as soon as the output has been checked, unless `TEST` is being run with the option `-keep`.

If most of the tests fail, it is likely that there’s something wrong with the installation. Look carefully through `Makefile.config`, and consider turning down or even off optimization.

If there are only a few tests that fail, and LSDALTON seems to exit normally in each case, there may just be some issues with numerical accuracy. Different machines give slightly different results, and while we’ve tried to allow for some slack in the tests, it may be that your machine yields numbers just outside the intervals we’ve specified as acceptable. A closer comparison of the results with numbers in the test script and/or the reference output files should reveal whether this is actually the case. If numerical (in)accuracy is the culprit, feel free to send your output file(s) to `dalton-admin@kjemi.uio.no` so that we can adjust the numerical intervals accordingly.

Chapter 2

Maintenance

2.1 New versions, patches

New versions will be announced on the DALTON homepage (`http://www.kjemi.uio.no/software/dalton/dalton.html` and the DALTON mailing-list (`dalton-users@kjemi.uio.no`).

In between releases, bug fixes will be distributed as patches to an existing version. New patches will be freely available from the DALTON homepage, and will be announced on the DALTON mailing list. Patches will normally be distributed in order to correct significant errors. In case of minor changes, explicit instructions on modifying the source file(s) may be given.

In general, users should not make local modifications in the source code, as this usually makes it much harder to incorporate the changes into subsequent versions. It is more convenient to use the C preprocessor code. Indeed, by judicious use of local `#define` variables (this is described in any book on C that also describes the C preprocessor) it is often possible to isolate local modifications completely, making it much easier to apply them to later releases.

2.2 Reporting bugs and user support

The LSDALTON program is distributed to the computational chemistry society with no obligations on the side of the authors. The authors thus take no responsibility for the performance of the code or for the correctness of the results. This distribution policy gives the authors no responsibility for problems experienced by the users when using the LSDALTON program.

A mailing-list — `dalton-users@kjemi.uio.no` — has been created for the purpose of communication among the authors and the users about new patches and releases, and communication between users of the program. The authors hope that the users will be able to help each others out when problems arise. All authors of the LSDALTON program also receive mail from the mailing-list, but we do not guarantee that any author will reply to requests posted to the mailing-list. By default, all users signing a license agreement form will be added to this mailing list. If more people use the same licensed version, these users may send mail to `dalton-admin@kjemi.uio.no` asking to be added to this list as well as indicating whose licensed version of the program they are using.

Bug reports are to be reported to `dalton-admin@kjemi.uio.no` and will be dealt with by one of the authors, although no responsibility on the promptness of this response is given. In general, serious bugs that have been reported and fixed will lead to a new patch of the program, distributed from the DALTON homepage.

Part II

DALTON User's Guide

Chapter 3

Getting started with LSDALTON

In this chapter we give an introduction to the two input files needed for doing a calculation with the LSDALTON program (MOLECULE.INP and DALTON.INP), as well as the shell script file that is supplied with the program for moving relevant files to the scratch-directory and back to the home directory after a calculation has completed. A couple of examples of input files are also provided. Finally, the different output files generated by the program are discussed.

3.1 The MOLECULE input file

Basically, the MOLECULE file contains the following information:

- The Cartesian coodinates for all atoms
- The basis set(s) to be used

Please note the following:

- Supplying the molecular geometry in the form of a Z-matrix is not supported by LSDALTON since this format is ill suited for large scale calculations.
- Molecular symmetry is not exploited by LSDALTON since the target systems of the program are large biomolecules, which hardly ever contains any symmetry.

There a two different types of molecule file, using either BASIS or ATOMBASIS. ATOMBASIS is relevant only if different basis sets on individual atoms are required. In the following, we describe these two types of MOLECULE files using simple examples.

3.1.1 BASIS

A BASIS MOLECULE file may look like this:

```
1     BASIS
2     cc-pVDZ Aux=Ahlrichs-Coulomb-Fit
3     water R(OH) = 0.95Aa , <HOH = 109 deg.
4     Distance in Aangstroms
5     Atomtypes=2 Angstrom
6     Charge=8.0 Atoms=1
7     O      0.00000   0.00000   0.00000
8     Charge=1.0 Atoms=2
9     H      0.55168   0.77340   0.00000
10    H      0.55168  -0.77340   0.00000
```

We now describe the MOLECULE file line by line:

- Line 1: The word "BASIS", indicating that the same basis set should be used for all atoms.
- Line 2: The name of the basis set, in this case cc-pVDZ. Optional: The name of the auxiliary basis set, in this case Ahlrichs-Coulomb-Fit (only referenced if density-fitting is requested in DALTON.INP)
- Lines 3-4: Comments (may be left blank)
- Line 5: **Atomtypes=** Number of different atoms - in the case of H_2O, there are two different atoms type, H and O. **Angstrom** If this keyword is omitted, the program will assume that coordinates are given in atomic units (Bohr). Other options in this line are: **Charge=** Molecular charge - assumed to be zero if omitted.
- Lines 6 and 8: A line of this type has to come before the Cartesian coordinates of each atom type, indicating a) The nuclear charge of the atom type, and b) The number of atoms of this type.

- Lines 7, 9-10: Cartesian coordinates of the atoms (in Angstrom, since this was specified in line 5).

3.1.2 ATOMBASIS

An ATOMBASIS MOLECULE file may look like this:

```
ATOMBASIS
water R(OH) = 0.95Aa , <HOH = 109 deg.
Distance in Aangstroms
Atomtypes=2 Angstrom
Charge=8.0 Atoms=1 Basis=cc-pVDZ Aux=Ahlrichs-Coulomb-Fit
O      0.00000   0.00000   0.00000
Charge=1.0 Atoms=2 Basis=cc-pVDZ Aux=Ahlrichs-Coulomb-Fit
H      0.55168   0.77340   0.00000
H      0.55168  -0.77340   0.00000
```

As seen from the example, there is little difference between the two formats. Line 2 in the BASIS file has been removed, and instead the basis sets (and optionally, the auxiliary basis sets) are now given for each atom. In this example they are the same, as it would probably not make much sense to use different basis sets for the atoms in water. This can make sense, though, for larger molecules. An example could be a biomocule containing an active site with one or more transition metal atoms. In may be useful to describe such an active site at a higher level of theory than the surroundings.

3.2 The DALTON.INP file

In this section, we show two typical examples of a DALTON.INP file, one for small molecules and one for large molecules. Many other examples of input files for requesting e.g. specific molecular properties may be found at `http://www.kjemi.uio.no/software/dalton/dalton.html`, and a complete reference manual of all keywords is found in Chapter 4.

In general, the DALTON.INP is divided in four different sections under the headlines:

- **INTEGRAL contain settings for the calculation of integrals (optional)

- **WAVE FUNCTIONS contains info about the wave function (e.g. HF/DFT) and settings for optimization of the wave function (mandatory)
- **OPTIMIZE contains settings for geometry optimization (optional). If this keyword is present in the input a full geometry optimization will be performed, otherwise a single-point calculation is carried out.
- **RESPONS contains info about requested properties (optional)

Each of these sections may contain subsections, indicated by a single asterisk. DALTON.INP should always end with *END OF INPUT.

A typical for DALTON.INP for a **small molecule** (say, less than 100 atoms) could look like:

```
**INTEGRAL
.NOJENGINE
.NOLINK
**WAVE FUNCTIONS
.HF
*DENSOPT
.RH
.DIIS
.CONVTHR
1.0D-6
**RESPONS
.NEXCIT
5
*END OF INPUT
```

Here, we have requested a Hartree-Fock optimization under **WAVE FUNCTION. The subsection *DENSOPT contains setting for how the density should be optimized. The .RH (.i.e. Roothaan-Hall) and .DIIS keywords requests a standard diagonalization combined with the DIIS scheme for convergence acceleration. With the keyword .CONVTHR (i.e. convergence threshold) it is requested that iterations are terminated when the Frobenius norm of the SCF gradient is smaller than 10^{-6}. Finally, we have under **RESPONS requested the calculation of the five lowest excitation energies with .NEXCIT (i.e. number of excitation energies). Note that under **INTEGRAL we have turned off the LinK and J-engine schemes for speeding up the calculation of the exact exchange and Coulomb contributions, respectively, since these are probably not needed for a small molecule.

A typical for DALTON.INP for a **large molecule** (say, more than 100 atoms) could look like:

```
**INTEGRAL
.DENSFIT
.RUNMM
*FMM
.SCREEN
 1e-10
.LMAX
 5
.TLMAX
 12
**WAVE FUNCTIONS
.DFT
 BP86
*DFT INPUT
.GRID3
.DFTTHR
1.d-6 0.d0 1.d-7 1.d-9
*DENSOPT
.START
 TRILEVEL
.ARH
.CONVDYN
 STANDARD
*END OF INPUT
```

Under **INTEGRAL, we have requested the use of density-fitting and FMM (the Fast Multipole Method) to speed up the calculation (the use of J-engine is default). Under **WAVE FUNCTIONS, we now request a DFT calculation using the BP86 exchange-correlation functional and specify an efficient integration grid. Since this is a calculation on a large molecule, we request the trilevel starting guess (atomic, valence, and then full optimization) and the Augmented Rothaan-Hall (ARH) method for density optimization. Note that ARH is default, so this keyword is actually unnecessary. ARH is more robust than the standard diagonalization/DIIS scheme, and linear-scaling if sparse-matrix algebra is employed. You may use sparse-matrix algebra by putting .CSR (i.e. Compressed-Sparse Row) under *DENSOPT (NB: requires linking to the MKL library).

Part III

DALTON Reference Manual

Chapter 4

List of LSDALTON keywords

In this chapter, we describe all keywords for the different sections of the DALTON.INP file. In general, DALTON.INP is divided in four different sections under the headlines:

- **INTEGRAL contain settings for the calculation of integrals (optional)
- **WAVE FUNCTIONS contains info about the wave function (e.g. HF/DFT) and settings for optimization of the wave function (mandatory)
- **OPTIMIZE contains settings for geometry optimization (optional)
- **RESPONS contains info about requested properties (optional)

Each of these sections may contain subsections, indicated by a single asterisk. DALTON.INP should always end with *END OF INPUT.

4.1 **INTEGRAL

This input module defines the way the integrals should be calculated.

`.DENSFIT` Use density-fitting. When using this keyword, an auxiliary basis set has to be specified in MOLECULE file.

`.RUNMM` Use the (Fast) Multipole Method.

`.AOPRINT`

`<Print level>`

Print the Atomic Orbital information. The higher the print level, the more information will be printed.

`.BASPRINT`

`<Print level>`

Print the basis set information. The higher the print level, the more information will be printed.

`.CARMOM`

`<Order of multipole moment>`

Calculate the cartesian multipole moment integrals to the order requested.

$$\int \chi_\mu(r_1)x^i y^j z^k \chi_\rho(r_2) dr_1 dr_2 \tag{4.1}$$

`.CART-E` Use cartesian E-coefficients instead of hermite E-coefficients for the McMurchie-Davidson integral-evaluation scheme [1] used in LSDALTON. The use of Cartesian E-coefficients is default in most integral programs. We however use hermite E-coefficients according to ref. [2].

`.ETUVIL` Build E-coefficients inside the integral loop, in order to reduced memory requirements. This will increase the workload!

`.FTUVMAXPRIM`

`<Maxprim>`

Sets the maximum number of primitives used in the FTUV batches in the J-engine algorithm.

`.INTPRINT`

`<Print level>`

Print information about the actual integral evaluation. The higher the print level, the more information will be printed.

`.NOJENGINE` Turn off J-engine algorithm (which is default) for the calculation of the Coulomb matrix (see Refs. [3] and [4]). Compute instead Coulomb-like contributions from the explicitly calculated integrals.

`.NOLINK` Turn off the LinK algorithm (which is default) for the calculation of the exchange matrix (see Ref. [5]).

`.LINSCAPRINT`

`<Print level>`

Print level in the integral code. The higher the print level, the more information will be printed.

`.LOW RJ000 ACCURACY` Use a decreased accuracy for the calculation of the Boys function.

`.MAXPASSES`

`<maxpass>`

Change the maximum number of collected overlaps (default is 40).

`.MOLPRINT`

`<Print level>`

Print information about the molecule. The higher the print level, the more information will be printed.

`.NO CS` Deactivate the use of Cauchy-Schwarz screening.

`.NO PASS` Deactivate the use of Passes, the collection of overlaps which is used as default in order to increase efficiency.

`.NO PS` Deactivate the use of primitive Cauchy-Schwarz screening.

`.NO SCREEN` Deactivate the use of all screening methods.

`.NOFAMILY` Deactivate the exploitation of family type basis set, basis set sharing exponents for different angular momentums.

`.NOSEGMENT` Deactivate the use of segments. The use of segments is used to reduced the numer of primitive functions in the calculation of integrals.

`.NSETUV` Use non-spherical E-coefficients.

`.OVERLAP-DF-J` Use the Overlap density fitting algorithm for the calculation of the Coulomb matrix.

`.SPHMOM`

`<Order of multipole moment>`

Calculate spherical multipole moment integrals to the requested order.

`.THR_CS`

`<CS_Threshold>`

Cauchy-Schwarz screening threshold.

`.THR_PS`

`<PS_Threshold>`

Primitive Cauchy-Schwarz screening threshold.

`.THRESH`

`<Threshold>`

An overall screening threshold for integral evaluation. The various integral-evaluation thresholds (below) are set according to

- The Cauchy-Schwarz screening threshold is set equal to Threshold (can be set separately by `.THR_CS`)
- The Primitve Cauchy-Schwarz screening threshold is set equal to Threshold$\cdot 10^{-1}$ (can be set separately by `.THR_PS`)
- The Overlap-distribution distance-screening threshold is set equal to Threshold$\cdot 10^{-1}$. When calculating overlap integrals, this threshold is used for setting up AO extents. Overlap distributions (ODs) for which the distance between the two AOs are larger than the sum of the extents are screened away.
- The Overlap-distribution extent-screeing threshold is set equal to Threshold$\cdot 10^{-1}$. When calculating overlap integrals, this threshold is used for setting up OD extents. Overlap integrals between two ODs separated by more than the sum of the extents are screened away.
- The FMM threshold is used to distinguish the Coulomb repulsion into classical and non-classical interactions, based on the non-classical extent of the contineous charge distribution (orbitals or orbital products). By default this threshold is set equal to Threshold$\cdot 10^{-1}$ (can be set separately by `.SCREEN` under *FMM)

`.UNCONT` Treat the basis functions (given by input) as fully uncontracted basis fuctions (i.e. ignore any contraction coefficients given in the basis and treat instead all the primitives as separate basis functions).

4.1.1 *FMM

This input block is used to specify settings for the (continuous) fast multipole moment (FMM) treatment of classical Coulomb interactions.
For gradients OpenMP parallelisation is turned off. The reason for this is that a continous counter used to identify moments for orbital products has to be the same in both moments and derivative moments. For the same reason screening [only during writing (see printmm routines)!!] is turned off both for the serial and parallel cases. As a consequence of the

screeening been turned off the moments have to be recalculated for the gradient and can not be reused from the preceeding energy evaluation. The gradient implementation has been tested both for regular and density fitting Coulomb interactions. It works for the combined N-e + e-e + N-N interactions (default), and for the e-e interaction only (invoked by `.NOONE`). It has however not been tested for speed and efficiency.

`.LMAX`

`<Lmax>`

The maximum multipole-moment expansion order of a given orbital product.

`.TLMAX`

`<TLmax>`

The maximum order used for translated multipole-moments.

`.SCREEN`

`<Threshold>`

Defines the threshold used to determine if a contribution can be calculated classically on non-classically.

`.NOONE` Do not use FMM for the nuclear-electron attraction.

`.NOMMBU` Do not use io-buffers for interfacing multipole-moments to the FMM driver.

4.2 **WAVE FUNCTION

`.HF` Hartree-Fock calculation.

`.DFT`

`<FUNCTIONAL>`

DFT calculation. The exchange—correlation functional is read from the following line.

Functionals in LSDALTON can be divided into two groups: generic and combined functionals. Combined functionals are a linear combination of generic ones. One can always create own combined functionals by using GGAKey general functional. A number of standalone functionals are also included within LSDALTON.

It should be noted that the input is not case sensitive, although the notation employed in this manual makes use of case to exphasise exchange or correlation functional properties and reflect the original literature sources.

Supported functionals are:

Exchange Functionals

`Slater` Dirac-Slater exchange functional [6, 7, 8].

`Becke` 1988 Becke exchange GGA correction [9]. Note that the full Becke88 exchange functional is given as Slater + Becke.

`KTx` Keal and Tozer's 2003 GGA exchange functional (With x being 1,2 or 3). Note that the gradient correction pre-factor constant, γ, is not included in the KT exchange definition, but rather in the KT1, KT2 and KT3 definitions. The full KT exchange is given by [10], Slater + γKTx (γ is -0.006 for KT1,KT2 and -0.004 for KT3).

`OPTX` Handy's 2001 exchange functional correction [11]. The full OPTX exchange functional is given by 1.05151*Slater - 1.43169*OPTX.

`PBEx` Perdew, Burke and Ernzerhof 1996 exchange functional [12].

`PW86x` Perdew and Wang 1986 exchange functional (the PWGGA-I functional)[13].

Correlation Functionals

`VWN3` correlation functional of Vosko, Wilk and Nusair, 1980 (equation III) [14]. This is the form used in the Gaussian program.

`VWN5` correlation functional of Vosko, Wilk and Nusair, 1980 (equation V – the recommended one). The VWN keyword is a synonym for VWN5 [14].

`LYP` correlation functional by Lee, Yang and Parr, 1988 [15, 16].

`P86c` non-local part of the correlation functional of the Perdew 1986 correlation functional [17]. PZ81 (1981 Perdew local) is usually used for the local part of the functional, with a total corelation functional of P86c + PZ81.

`PBEc` Perdew, Burke and Ernzerhof 1996 correlation functional, defined as PW91c local and PBEc non-local correlation [12].

`PW91c` 1991 correlation functional of Perdew and Wang (the pwGGA-II functional) [18]. This functional includes both the PW91c non-local and PW91c local (ie PW92c) contributions. The non-local PW91c contribution may be determined as PW91c - PW92c.

`PZ81` local correlation functional of Perdew and Zunger, 1981 [19].

Standalone Functionals

`LB94` asymptotically correct functional of Leeuwen and Baerends 1994 [20]. This functional improves description of the asymptotic density on the expense of core and inner valence.

Combined functionals

`Combine` is a universal keyword allowing users to manually construct arbitrary linear combinations of exchange and correlation functionals from the list above. Even fractional Hartree–Fock exchange can be specified. This keyword is to be followed by a string of functionals with associated weights. The syntax is `NAME=WEIGHT ...`. As an example, B3LYP may be constructed as:

```
.DFT
 Combine HF=0.2 Slater=0.8 Becke=0.72 LYP=0.81 VWN=0.19
```

The following GGA and hybrid functional aliases are defined within DALTON and provide further examples of the Combine keyword.

`SVWN5` is a sum of Slater functional and VWN (or VWN5) correlation functional. SVWN is a synonym for SVWN5. It is equivalent to
`Combine Slater=1 VWN5=1`

`SVWN3` is a sum of the Slater exchange functional and VWN3 correlation functional. It is equivalent to the Gaussian program LSDA functional and can alternatively be selected by following set of keywords
`Combine Slater=1 VWN3=1`

`LDA` A synonym for SVWN5 (or SVWN).

`BLYP` is a sum of Slater functional, Becke88 correction and LYP correlation functional. It is equivalent to
`Combine Slater=1 Becke=1 LYP=1`

`B3LYP` 3-parameter hybrid functional [21] equivalent to:
`Combine HF=0.2 Slater=0.8 Becke=0.72 LYP=0.81 VWN=0.19`

`B3LYP-G` hybrid functional with VWN3 form used for correlation—this is the form used by the Gaussian quantum chemistry program. Keyword B3LYPGauss is a synonym for B3LYPg. This functional can be explicitely set up by
`Combine HF=0.2 Slater=0.8 Becke=0.72 LYP=0.81 VWN3=0.19`

`BP86` Becke88 exchange functional and Perdew86 correlation functional (with Perdew81 local correlation). The explicit form is:
`Combine Slater=1 Becke=1 PZ81=1 P86c=1`

`B3P86` variant of `B3LYP` with VWN used for local correlation and P86 for the nonlocal part.
`Combine HF=0.2 Slater=0.8 Becke=0.72 P86c=0.81 VWN=1`

`B3P86-G` variant of `B3LYP` with VWN3 used for local correlation and P86 for the nonlocal part. This is the form used by the Gaussian quantum chemistry program.
`Combine HF=0.2 Slater=0.8 Becke=0.72 P86c=0.81 VWN3=1`

`BPW91` Becke88 exchange functional and PW91 correlation functional. The explicit form is:
`Combine Slater=1 Becke=1 PW91c=1`

`CAMB3LYP` Coulomb Attenuated Method Functional of Yanai, Tew and Handy [22]. This functional accepts additional arguments `alpha`, `beta` and `mu` to modify the fraction of HF exchange for short-range interactions, additional fraction of HF exchange for long-range interaction and the interaction switching factor μ. This input can be specified as follows:

```
.DFT
 CAMB3LYP alpha=0.190 beta=0.460 mu=0.330
```

`KT1` Slater-VWN5 functional with the KT GGA exchange correction [10, 23].
`Combine Slater=1 VWN=1 KT=-0.006`

`KT2` differs from KT1 only in that the weights of the Slater and VWN5 functionals are from an empirical fit (not equal to 1.0) [10, 23].
`Combine Slater=1.07173 VWN=0.576727 KT=-0.006`

`KT3` a hybrid functional of Slater, OPTX and KT exchange with the LYP correlation functional [24]. The explicit form is
`Combine Slater=1.092 KT=-0.004 LYP=0.864409 OPTX=-0.925452`

`OLYP` is the sum of the OPTX exchange functional with the LYP correlation functional [11, 15, 16].
`Combine Slater=1.05151 OPTX=-1.43169 LYP=1`

`PBE0` a hybrid functional of Perdew, Burke and Ernzerhof with 0.25 weight of exact exchange, 0.75 of `PBEx` exchange functional and the `PBEc` correlation functional [25]. Alternative aliases are PBE1PBE or PBE0PBE.
`Combine HF=0.25 PBEx=0.75 PBEc=1`

`PBE` same as above but with exchange estimated exclusively by PBEx functional [12]. Alias of PBEPBE. This is the form used by CADPAC and NWChem quantum chemistry programs.
`Combine PBEx=1 PBEc=1`
Note that the Molpro quantum chemistry program uses the PW91c non-local correlation functional instead of PBEc, which is equivalent to the following:
`Combine PBEx=1 PW91c=1` .

Note that combinations of local and non-local correlation functionals can also be generated with the Combine keyword. For example, `Combine P86c=1 PZ81=1` combines the PZ81 local and P86c non-local correlation functional, whereas `Combine VWN=1 P86c=1` combines the VWN local and P86 non-local correlation functionals.

Linear combinations of all exchange and correlation functionals listed above are possible with the `Combine` keyword.

4.2.1 *DFT INPUT

Controls the XC-integration grid. The grid is generated as follows: For each atom a radial grid is created (different radial grids available, see below), which is multiplied with an angular grid (Lebedev-grids). The angular grids are by default pruned for radial points close to the nucleus, following Murray, Handy and Laming. [26] Grid weights for the grid points are calculated following different schemes (see below). Grid points with weights below 10^{-20} are disregarded. Please note that the grid construction routines in the LSDALTON program changed compared to the main version of DALTON which may result in small deviations of the number of grid points etc.

`.ANGINT`

`<ANGINT>`

Determines the quality of the angular Lebedev grid – the angular integration of spherical harmonics will be exact up to the specified order. Default value is 31. Maximum value is 64. Note that the value of `ANGINT` is changed by the following keywords: `COARSE`, `NORMAL`, `FINE`, `ULTRAFINE`. The `GRIDX` keywords also imply specific `ANGINT` values.

`.COARSE` Shortcut keyword for radial integration accuracy 10^{-11} and angular expansion order equal to 35.

`.DFTELS`

`<DFTELS>`

safety threshold – stop if the charge integration error becomes larger then this threshold.

`.DFTTHR`

`<DFTHR0, DFTHRL, DFTHRI, RHOTHR>`

`DFTHR0`	is not used
`DFTHRL`	is not used
`DFTHRI`	threshold for screening product of gaussian atomic orbitals
`RHOTHR`	threshold for screening of the electron density

`.DISPER` Activates the addition of the empirical dispersion correction following S. Grimme. [27, 28] The S6 factors and van der Waals radii of Ref. [28] are used. Implemented for the functionals BP86, BLYP, PBE, B3LYP for energies and gradients. Does not work when functionals are defined with the `combine` command.

`.FINE` Shortcut keyword for radial integration accuracy 10^{-13} and angular expansion order equal to 42.

`.GRID TYPE`

`<GC2 LMG TURBO BECKE BECKEORIG SSF BLOCK BLOCKSSF>`

`GC2`, `LMG`, `TURBO` define radial quadrature schemes, only one can be set. `BECKE`, `BECKEORIG`, `SSF`, `BLOCK`, `BLOCKSSF` define the grid partitioning schemes, only one can be set.

`GC2` Gauss-Chebyshev quadrature of second kind.

`LMG` As proposed by Lindh, Malmqvist and Gagliardi (default). [29]

`TURBO` Treutler-Ahlrichs M4-T2 scheme. [30] Implies also that the angular integration quality becomes Z-dependant (see also `ANGINT`) and that pruning is used (see also `NOPRUN`).

`BECKE` Becke partitioning scheme with atomic size correction. [31]

`BECKEORIG` Becke partitioning scheme without atomic size correction (default). [31]

`SSF` Stratmann-Scuseria-Frisch partitioning scheme. [32]

`BLOCK` Becke partitioning scheme with atomic size correction [31] combined with a blockwise handling of grid points [33]. Useful for large molecules.

`BLOCKSSF` Stratmann-Scuseria-Frisch partitioning scheme [32] combined with a blockwise handling of grid points [33]. Useful for large molecules.

`.GRID1` Shortcut for radial integration accuracy $1.0 * 10^{-5}$, angular expansion order equal to 17, radial quadrature `TURBO`, grid partitioning scheme `BLOCK`.

`.GRID2` Shortcut for radial integration accuracy $2.15447 * 10^{-7}$, angular expansion order equal to 23, radial quadrature `TURBO`, grid partitioning scheme `BLOCK`.

`.GRID3` Shortcut for radial integration accuracy $4.64159 * 10^{-9}$, angular expansion order equal to 29, radial quadrature `TURBO`, grid partitioning scheme `BLOCK`.

`.GRID4` Shortcut for radial integration accuracy $5.01187 * 10^{-14}$, angular expansion order equal to 35, radial quadrature `TURBO`, grid partitioning scheme `BLOCK`.

`.GRID5` Shortcut for radial integration accuracy $2.15443 * 10^{-17}$, angular expansion order equal to 47, radial quadrature `TURBO`, grid partitioning scheme `BLOCK`.

`.HARDNESS`
: `<HARDNESS>`
sets the hardness of the partitioning function in the Becke weighting scheme. Only positive integer values allowed. The higher the hardness the more stepfunction like the partitioning functions. Default value is 3 as proposed by Becke. [31]

`.NOPRUN` Supresses the default pruning of the atomic angular integration grids for radial points close to the nucleus.

`.NORMAL` Shortcut keyword for radial integration accuracy 10^{-13} and angular expansion order equal to 35.

`.RADINT`
: `<RADINT>`
Determines the quality of the radial integration grid. Default radial integration accuracy is 10^{-11}. Note that the value of `RADINT` is changed by the following keywords: `NORMAL`, `FINE`, `ULTRAFINE`. The `GRIDX` keywords also imply specific `RADINT` values.

`.ULTRAF` Shortcut keyword for radial integration accuracy 10^{-15} and angular expansion order equal to 64.

4.2.2 *DENSOPT

This section contains the different options for obtaining the SCF wave function and energy. The default is diagonalization combined with the DIIS scheme for acceleration of SCF convergence (no damping/level shifting). Should have a different name. DENSOPT?

`.2ND_ALL` Use second order optimization in all SCF iterations.

`.2ND_LOC` Use second order optimization in local SCF iterations.

`.ARH` Use Augmented Roothaan-Hall scheme for density optimization (default) [34, 35].

`.ARH FULL` Use Augmented Roothaan-Hall for density optimization - no truncation of the reduced space, keep all micro vectors. This is recommended if the standard scheme (.ARH) fails to converge (see ref. [36]).

`.ASYM` Asymmetrize H1DIAG starting guess (only referenced for unrestricted calculations).

`.STABILITY` Check stability of the optimized wave function by calculation of the lowest Hessian eigenvalue.

`.STAB_MAXIT`
: `<Max. number of stability iterations>`
Max. number of iterations in calculation of lowest Hessian eigenvalue (default is 40).

`.CHOLESKY` Do Cholesky decomposition of overlap matrix (for density optimization in orthogonal AO basis) - default is Löwdin decomposition by diagonalization.

`.CONTFAC`
: `<Contraction factor>`
For update of trust-radius (used in ARH and second order optimization). If trust-radius should be contracted, contract it by this factor (default is 0.7).

`.CONTRAC`
: `<Contraction criterion>`
For update of trust-radius (used in ARH and second order optimization). Contract trust-radius if trust-radius ratio is smaller than this criterion (default is 0.25).

`.CONVDYN`
: `<Option>`
Dynamic SCF convergence threshold. This is suitable for large calculations, since the standard SCF convergence threshold is based on the Frobenius norm of the SCF gradient, which is not size-extensive. Options are TIGHT, STANDARD, and SLOPPY.

`.CONVTHR`
: `<Threshold>`
SCF convergence threshold - Frobenius norm of SCF gradient (default is 1.0d-4). Note that this convergence criterion is not size-extensive. For large molecules, it is recommended to use .CONVDYN instead.

`.CSR` Use Compressed-Sparse Row matrices. NB: Requires linking to MKL library!

`.DIIS` Pulay's DIIS scheme is used to speed up SCF convergence [37, 38].

`.DISK` Store densities and Fock/KS matrices from previous iterations on disk instead of in core when constructing the new Fock/KS matrix (*memory-saving option*).

`.DISKSOLVER` Store trial and sigma vectors on disk instead of in core when solving linear equations (*memory-saving option*). Only referenced is density optimization method is ARH, Direct Density optimization or second order optimization.

`.DORTH` Do level shift by using the ratio ||Dorth||/||D|| (for diagonalization only).

`.DSM` The Density Subspace Minimization scheme is used to speed up SCF convergence [39, 40].

`.DSMONE` The Density Subspace Minimization scheme is used to speed up SCF convergence, but only one DSM step is taken in each SCF iteration.

`.DSMXTRA` The Density Subspace Minimization scheme is used to speed up SCF convergence, including extra (more expensive) term.

`.DUMPMAT` Dump Fock/KS and density matrices from all iterations to disk for later investigation.

`.EDIIS` The energy-DIIS scheme is used to speed up SCF convergence [41].

`.EXPAND`

`<Expansion criterion>`

For update of trust-radius (used in ARH and second order optimization). Expand trust-radius if trust-radius ratio is larger than this criterion (default is 0.75).

`.EXPFAC`

`<Expansion factor>`

For update of trust-radius (used in ARH and second order optimization). If trust-radius should be expanded, expand it by this factor (default is 1.2).

`.FIXSHIFT`

`<Shift>`

Use a fixed level shift in all SCF iterations.

`.HESVEC`

`<Number of Hessian eigenvalues>`

Number of lowest Hessian eigenvalues to be calculated with keyword .STABILITY (default is 1).

`.LCV` Compute the Least-Change Valence orbitals after valence density optimization step. All subsequent calculations are computed using Least-Change Valence orbitals augmented with atomic virtual orbitals of the Atomic density as basis set. Keyword only effective with TRILEVEL starting guess.

`.LCM` Compute the Least-Change Molecular orbitals after the Full molecular density calculation. All subsequent calculations are computed using these orbitals as basis set. .LCV is automatically included. Keyword only meaningful with TRILEVEL starting guess.

`.LEVELSH`

`<N> <shift1> <shift2> ...<shiftN>`

Use custom level shifts in the first N SCF iterations.

`.LWITER` Do Löwdin decomposition of overlap matrix (for density optimization in orthogonal AO basis) - iteratively (default is Löwdin by diagonalization) [42].

`.LWQITER` Do Löwdin decomposition of overlap matrix in quadruple precision (for density optimization in orthogonal AO basis) - iteratively.

`.MAXELM`

`<Trust radius (max. element)>`
Absolute max. element of step allowed (default is 0.35). Used in ARH and second order optimization.

`.MAXIT`

`<Max. number of iterations>`
Max. number of SCF iterations (default is 100).

`.MAXRATIO`

`<Max. ratio for DORTH>`
Largest accepted ratio when using level shift .DORTH.

`.MAXSTEP`

`<Trust radius (Frobenius norm)>`
Absolute max. Frobenius norm of step allowed (default is 0.6). Used in ARH and second order optimization.

`.MICTHRS`

`<Threshold for micro iterations>`
The micro iterations are converged to gradient norm times this factor (default is 1.0d-2). Only referenced if ARH, Direct Density optimization or second order optimization.

`.MICROVECS`

`<Max. number of microvectors>`
Max. number of microvectors to be kept in Conjugate Residual Optimal Vectors (CROP) scheme (default is 2). Only referenced if ARH, Direct Density optimization or second order optimization.

`.MINDAMP`

`<Minimum damping>`
Never allow level shift to be smaller than this.

`.MOCHANGE` Level shifting is done by use of MO overlaps (diagonalization only).

`.MUOPT` Optimal level shift is chosen by doing a line search in the SCF energy - very expensive! For diagonalization only.

`.NALPHA`

`<Number of alpha electrons>`

Explicitly specify number of alpha electrons (automatically set if not specified). If both NALPHA and NBETA are set, they must conform with the total number of electrons!

`.NBETA`

`<Number of beta electrons>`

Explicitly specify number of alpha electrons (automatically set if not specified). If both NALPHA and NBETA are set, they must conform with the total number of electrons!

`.NOAV` Turn off Fock matrix averaging in SCF iterations (this is default since the default optimization scheme is ARH, which contains implicit averaging).

`.NVEC`

`<Max. no of vectors for averaging>`

Maximum number of previous Fock and density matrices to be stored and used for ARH, DIIS, and DSM (default is 10 for HF and 7 for DFT).

`.NOINCREM` Turn off incremental scheme for Fock/KS matrix build.

`.NOPREC` Turn off preconditioning of linear equations (ARH, TrFD, and second order optimization).

`.NOSHIFT` Do no level shifting.

`.OAO` Keep the whole calculation (also Fock/KS matrix construction) in orthonormal AO basis.

`.OVERLAP`

`<Overlap>`

Smallest accepted overlap when doing level shift by use of MO overlaps (.MOCHANGE).

`.RH` Use standard Roothaan-Hall scheme (diagonalization) for density optimization.

`.START`

`<Option>`

Starting guess for SCF optimization. Options are:

- **H1DIAG** Obtain initial guess by diagonalizing one-electron Hamiltonian
- **H1OAO** Obtain initial guess by diagonalizing one-electron Hamiltonian in orthonormal AO basis

- **ATOMS** Obtain initial guess by atoms-in-molecules approach (default)
- **TRILEVEL** Optimization in three steps. 1. Atomic densities, 2. Valence molecular density, 3. Full molecular density [43, 44].

Note that Huckel guess is not supported by LSDALTON.

`.TRSCF` Use Trust-Region SCF scheme, i.e. diagonalization, .DORTH level shift, and DSM [39, 40].

`.TrFD` Density optimization by minimization of the Roothaan-Hall energy $E_{RH} = \mathrm{Tr}\mathbf{FD}(\mathbf{X})$ [45].

`.TrFD FULL` As above, but keep the full subspace of trial vectors (instead of truncating) when solving linear equations.

`.UNREST` Force unrestricted calculation (default for open shell systems).

`.VanLenthe` Use Van Lenthe's scheme for level shifting and averaging [46].

4.2.3 $INFO

For the different parts of the wave function optimization, it is possible to get very detailed information. This section describes these keywords. They must be put under *LINSCA in a section beginning with $INFO and ending with $END INFO. Note that these keywords do NOT start with a dot!

`INFO_CROP` Detailed info from Conjugated Residual OPtimal vectors scheme (Direct Density Optimization, ARH and second order optimization).

`INFO_DIIS` Detailed info from Direct Inversion in the Iterative Subspace algorithm.

`INFO_DSM` Detailed info from Density Subspace Minimization algorithm.

`INFO_DSM_DETAIL` Even more detailed info from Density Subspace Minimization algorithm.

`INFO_LEVELSHIFT` Detailed info from determination of dynamic level shift (Direct Density Optimization, ARH and second order optimization).

`INFO_LINEQ` Detailed info from solution of linear equations and trust-radius update (Direct Density Optimization, ARH and second order optimization).

`INFO_RH` Detailed info from Roothaan-Hall algorithm (diagonalization).

`INFO_RH_DETAIL` Even more detailed info from Roothaan-Hall algorithm (diagonalization).

`INFO_STABILITY` Detailed info from calculation of lowest Hessian eigenvalue (stability analysis) and HOMO-LUMO gap.

`INFO_STABILITY_REDSPACE` Reduced space info from calculation of lowest Hessian eigenvalue (stability analysis) and HOMO-LUMO gap.

`INFO_WEIGHT` Detailed info about weights of the densities in DIIS and DSM algorithms.

4.3 **OPTIMIZE

This input module describes keywords needed to optimize molecular geometry. In the current release a number of quasi-Newton trust-radius level-shifted techniques for finding equilibrium geometries are available. The methods differ in the way approximate Hessians are built and updated. Geometric structure can be optimized in redundant internal or Cartesian coordinates with two sets of convergence criteria. The default setting is minimization in internal coordinates [47], with model Hessian [48] updated with BFGS formula. By default, new convergence criteria are used, i.e. convergence is declared if the four quantities - the root-mean-square of the gradient, its maximum absolute value, the root-mean square of the step vector, and its maximum absolute element (in internal or Cartesian coordinates) - are smaller than ϵ, 5ϵ, 3ϵ and 15ϵ respectively [49]. The default value of ϵ is $1.0 \cdot 10^{-5}$.

By default, the atoms-in-molecule approach is used to set up the initial density-matrix at each new geometry (each step). However, it is possible to take as starting guess McWeeny-purified converged density-matrix at the previous geometry. This is achieved by setting the .RESTART option under the *DENSOPT subsection.

`.PRINT`

`<Print level>`

Print level for the output. The higher the print level, the more information is printed to the output file.

`KeyVLOOSE` For the default convergence criteria ϵ is set to $1.0 \cdot 10^{-3}$.

`KeyLOOSE` For the default convergence criteria ϵ is set to $1.0 \cdot 10^{-4}$.

`KeyTIGHT` For the default convergence criteria ϵ is set to $1.0 \cdot 10^{-6}$.

`KeyVTIGHT` For the default convergence criteria ϵ is set to $1.0 \cdot 10^{-7}$.

`.BAKER` Activates the convergence criteria of Baker [50]. The minimum is then said to be found when the largest element of the gradient vector (in Cartesian or redundant internal coordinates) falls below $3.0 \cdot 10^{-4}$ and either the energy change from the last iteration is less than $1.0 \cdot 10^{-6}$ or the largest element of the predicted step vector is less $3.0 \cdot 10^{-4}$.

`.BFGS` Specifies the use of a first-order method with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update formula for optimization. This is the preferred first-order method for minimizations, as this update is able to maintain a positive definite Hessian.

`.PSB` Specifies that a first-order method with the Powell-Symmetric-Broyden (PSB) update formula should be used for optimization.

`.DFP` Specifies that a first-order method with the Davidon-Fletcher-Powell (DFP) update formula should be used for optimization.

`.INIMOD` Use a simple model Hessian [48] diagonal in redundant internal coordinates as the initial Hessian. All diagonal elements are determined based on an extremely simplified molecular mechanics model, yet this model provides Hessians that are good starting points for most systems, thus avoiding any calculation of the exact Hessian. This is the default for optimizations in redundant internal coordinates. Currently this is not an option for optimization in Cartesian coordinates.

`.INIRED` Specifies that the initial Hessian should be diagonal in redundant internal coordinates. The different diagonal elements are set equal to 0.5 for bonds, 0.2 for angles and 0.1 for dihedral angles, unless `.INITEV` has been specified. If the optimization is run in Cartesian coordinates, the diagonal internal Hessian is transformed to Cartesians.

`.MODHES` Determine a new model Hessian (see `.INIMOD`) at every geometry without doing any updating. The model is thus used in much the same manner as an exact Hessian, though it is obviously only a relatively crude approximation to the analytical Hessian.

`.INITEV`

```
<Read EVLINI, the eigenvalues>
```

The default initial Hessian for first-order minimizations is the identity matrix when Cartesian coordinates are used, and a diagonal matrix when redundant internal coordinates are used. If `.INITEV` is used, all the diagonal elements (and therefore the eigenvalues) are set equal to the value EVLINI.

`.REDINT` Specifies that redundant internal coordinates coordinates should be used in the optimization. This is the default.

`.CARTES` Indicates that Cartesian coordinates should be used in the optimization.

4.4 **RESPONS

This section describes the keywords needed for obtaining molecular properties. The response section begins with **RESPONS. General response information related to more than one property (how many excitation energies, threshold for the response solver etc.) is put directly under **RESPONS. After this, each individual response property (e.g. the polarizability tensor) is labeled by an asterisk, and the specific (optional) information related to that property (e.g. which optical frequencies) follows below. All input values are given in atomic units. The general structure is thus:

```
**RESPONS

.General response information

*Label for property 1

.Specific (optional) information for property 1

*Label for property 2

.Specific (optional) information for property 2
```

Each of the properties below have a test case. The test cases may be found in the **test/** directory. To run a test case, go to the **test/** directory and type:

```
 ./TEST  <test case>
```

The test case files contains simple examples of full input files.

Below, the list of general response input keywords follows.

`.NEXCIT`

`<Number of excitation energies>`
Determines the number of excitation energies to be calculated. If this keyword is set the excitation energies and the corresponding one-photon dipole absorption strengths for excited states 1 to NEXCIT are always calculated. **Important:** If properties involving excited states are requested (e.g. two-photon absorption or the excited state gradient) this keyword must be listed *before* any of these properties.
Test case: LSresponse/LSresponse_HF_opa

`.*SOLVER`

`.<Solver information>`

General information related to the way response equations are solved to obtained property. By default response equations are solved using the algorithm were trial vectors are added in pairs.

`.NEW_SOLVER`

The new solver with symmetrized trial vectors is used for solving response equations.

`.COMPLEX`

`<Damping parmeter gamma>`

The γ parmeter is treated as an imaginary component of a frequency of intrest and describes a broadening of a spectrum.

`.OLSEN`

The Olsen algorithm is used for obtaining excitation energies. The Olsen algorithm works under the algorithm with symmetrized trial vectors.

`.CONVTHR`

`<Convergence threshold>`

Convergence threshold to which response equations are solved. Default value is set equal to 10^{-4}.

`.MAXIT`

`<Maximum number of iterations>`

Maximum number of iterations in the iterative procedure. Default value set equal to 100.

`.AOPREC`

AO preconditioning is used (MO preconditioning by default). AO preconditioning may only be used within the algorithm with paired trial vectors.

`.NOPREC`

No precondtioning is used.

`.CONVDYN`

`<Option>`

Dynamic convergence threshold suitable for large calculations. Options are TIGHT, STANDARD, and SLOPPY.

Now follows a list of input keywords for the specific properties available in LSDALTON.

4.4.1 *ALPHA

Calculation of the (possibly complex) polarizability tensor α and its isotropic average. The polarizability equals minus the linear response function $<< \mu_A; \mu_B >>_{\omega_B}$, where ω_B is in general allowed to be complex – i.e. $\omega_B = \omega_B^R + i\omega_B^I$. Frequencies not specified are set to zero by default, i.e. if no frequencies are specified, the static polarizability is calculated.
Test case: LSresponse/LSresponse_HF_alpha

`.BFREQ`

`<Number of real frequencies for B operator>`
`<Freq1, freq2, ... , freqN>`
Real frequencies ω_B^R in the corresponding linear response function. First line after .BFREQ contains the number of frequencies and the second line contains all frequencies on one line.

`.IMBFREQ`

`<Number of imaginary frequencies for B operator>`
`<Freq1, freq2, ... , freqN>`
Imaginary frequencies ω_B^I in the corresponding linear response function. The imaginary part of the complex polarizability describes a broadened one-photon absorption spectrum. If .IMBFREQ is not specified, only the real polarizability is calculated. Input as for .BFREQ.

For example, the following input is used to calculate complex polarizability tensors for the frequencies $\omega_B = 0.1 + 0.05i$ and $\omega_B = 0.2$ (in a.u.):

```
**RESPONS

*ALPHA

.BFREQ

2

0.1 0.2

.IMBFREQ

2

0.05 0.0
```

4.4.2 *BETA

Calculation of the (possibly complex) first hyperpolarizability tensor β and the corresponding averages $\beta_{||}$ and $\beta_{\perp}$. The first hyperpolarizability tensor equals minus the quadratic response function $<< \mu_A; \mu_B, \mu_C >>_{\omega_B,\omega_C}$, where the frequencies are in general allowed to be complex – i.e. $\omega_B = \omega_B^R + i\omega_B^I$ and $\omega_C = \omega_C^R + i\omega_C^I$. The input is analogous to the *ALPHA input. Frequencies not specified are set to zero by default, i.e. if no frequencies are specified, the static first hyperpolarizability is calculated.
Test case: LSresponse/LSresponse_HF_beta

```
.BFREQ
```

```
<Number of real frequencies for B operator>
<Freq1, freq2, ... , freqN>
```

Real frequencies ω_B^R in the corresponding quadratic response function. Input as for .BFREQ under *ALPHA.

```
.IMBFREQ
```

```
<Number of imaginary frequencies for B operator>
<Freq1, freq2, ... , freqN>
```

Imaginary frequencies ω_B^I in the corresponding quadratic response function. Input as for .BFREQ under *ALPHA.

```
.CFREQ
```

```
<Number of real frequencies for C operator>
<Freq1, freq2, ... , freqN>
```

Real frequencies ω_C^R in the corresponding quadratic response function. Input as for .BFREQ under *ALPHA.

```
.IMCFREQ
```

```
<Number of imaginary frequencies for C operator>
<Freq1, freq2, ... , freqN>
```

Imaginary frequencies ω_C^I in the corresponding quadratic response function. Input as for .BFREQ under *ALPHA.

4.4.3 *DAMPED_TPA

Damped two-photon absorption where both individual photons have the same energy (= half the excitation energy).
Test case: LSresponse/LSresponse_HF_dtpa

```
.OPFREQ
     <Number of one-photon frequencies>
     <Freq1, freq2, ... , freqN>
```
The frequencies in the input are one-photon frequencies (=half the excitation energy in case of resonance).

```
.GAMMA
     <Damping parameter>
```
The damping parameter γ is used in *all* response equations, i.e. $i\gamma$ is effectively added to all frequencies occuring in response equations. Default value: 0.005 a.u.

4.4.4 *ESDIPOLE

Permanent dipole moment for excited states. **Always requires specification of .NEXCIT!** If .EXSTATES is not specified, the excited state dipole moments for all excited states from 1 to NEXCIT are calculated.
Test case: LSresponse/LSresponse_HF_esd

```
.EXSTATES
     <Number of specific excited states to consider>
     <Specific excited states to consider>
```
Only calculate excited state gradient for selected excited states. The first line after .EXSTATES contains the number of excited states where the excited state gradient is to be calculated, and the second line specifies which states. **Important:** The position of the highlest lying excited state specified by .EXSTATES must be smaller than or equal to the number of excited states specified by .NEXCIT!

4.4.5 *ESGRAD

Calculation of the molecular gradient for excited states. **Always requires specification of .NEXCIT!** If .EXSTATES is not specified, excited state gradients for all excited states from 1 to NEXCIT are calculated.
Test case: LSresponse/LSresponse_HF_esg

```
.EXSTATES
     <Number of specific excited states to consider>
     <Specific excited states to consider>
```
Input is identical to .EXSTATES under *ESDIPOLE.

For example, the following input is used to calculate excited state gradients for excited states number 3 and 6:

```
**RESPONS

.NEXCIT

6

*ESGRAD

.EXSTATES

2

3 6
```

4.4.6 *GAMMA

Calculation of the (possibly complex) second hyperpolarizability tensor γ and the corresponding averages $\gamma_{||}$ and $\gamma_{\perp}$. The second hyperpolarizability tensor equals minus the cubic response function $<< \mu_A; \mu_B, \mu_C, \mu_D >>_{\omega_B,\omega_C,\omega_D}$, where the frequencies are in general allowed to be complex – i.e. $\omega_B = \omega_B^R + i\omega_B^I$, $\omega_C = \omega_C^R + i\omega_C^I$, and $\omega_D = \omega_D^R + i\omega_D^I$. The input is analogous to the *ALPHA and *BETA inputs. Frequencies not specified are set to zero by default, i.e. if no frequencies are specified the static second hyperpolarizability is calculated.
Test case: LSresponse/LSresponse_HF_gamma

```
.BFREQ
     <Number of real frequencies for B operator>
     <Freq1, freq2, ... , freqN>
```
Real frequencies ω_B^R in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

```
.IMBFREQ
     <Number of imaginary frequencies for B operator>
     <Freq1, freq2, ... , freqN>
```
Imaginary frequencies ω_B^I in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

`.CFREQ`

`<Number of real frequencies for C operator>`

`<Freq1, freq2, ... , freqN>`

Real frequencies ω_C^R in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

`.IMCFREQ`

`<Number of imaginary frequencies for C operator>`

`<Freq1, freq2, ... , freqN>`

Imaginary frequencies ω_C^I in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

`.DFREQ`

`<Number of real frequencies for D operator>`

`<Freq1, freq2, ... , freqN>`

Real frequencies ω_D^R in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

`.IMDFREQ`

`<Number of imaginary frequencies for D operator>`

`<Freq1, freq2, ... , freqN>`

Imaginary frequencies ω_D^I in the corresponding cubic response function. Input as for .BFREQ under *ALPHA.

4.4.7 *MOLGRA

Single point calculation of the molecular gradient.
Test case: LSresponse/LSresponse_HF_molgra

4.4.8 *TPA

Two-photon absorption where both individual photons have the same energy (= half the excitation energy). **Always requires specification of .NEXCIT!** If .EXSTATES is not specified, two-photon absorption for all excited states from 1 to NEXCIT are calculated.
Test case: LSresponse/LSresponse_HF_tpa

`.EXSTATES`

`<Number of specific excited states to consider>`

`<Specific excited states to consider>`
Input is identical to .EXSTATES under *ESDIPOLE.

Bibliography

[1] L E McMurchie and E R Davidson. One- and two-electron integrals over cartesian gaussian functions. *J. Comp. Phys.*, 26:218–231, 1978.

[2] Simen Reine, Erik Tellgren, and Trygve Helgaker. A unified scheme for the calculation of differentiated and undifferentiated molecular integrals over solid-harmonic gaussians. *Phys. Chem. Chem. Phys.*, 9:4771–4779, 2007.

[3] Yihan Shao and Martin Head-Gordon. An improved j matrix engine for density functional theory calculations. *Chem. Phys. Lett.*, 323(5-6):425, 2000.

[4] Yihan Shao, Christopher A. White, and Martin Head-Gordon. Efficient evaluation of the coulomb force in density-functional theory calculations. *J. Chem. Phys.*, 114(15):6572, 2001.

[5] Christian Ochsenfeld, Christopher A. White, and Martin Head-Gordon. Linear and sublinear scaling formation of hartree–fock-type exchange matrices. *J. Chem. Phys.*, 109(5):1663, 1998.

[6] P. Hohenberg and W. Kohn. *Phys. Rev.*, 136:B864, 1964.

[7] W. Kohn and L. J. Sham. *Phys. Rev.*, 140:A1133, 1965.

[8] J. C. Slater. *Quantum Theory of Molecular and Solids*, volume 4, chapter The Self-Consistend Field for Molecular and Solids. McGraw-Hill, New York, 1974.

[9] A. D. Becke. *Phys. Rev. A*, 38:3098, 1988.

[10] T. W. Keal and D. J. Tozer. The exchange-correlation potential in kohn-sham nuclear magnetic resonance shielding calculations. *J. Chem. Phys.*, 119:3015, 2003.

[11] N. C. Handy and A. J Cohen. *Mol. Phys.*, 99:403, 2001.

[12] J. P. Perdew, K. Burke, and M. Ernzerhof. *Phys. Rev. Lett.*, 77(3865), 1996.

[13] J. P. Perdew and Y. Wang. *Phys. Rev. B*, 33:8800, 1986.

[14] S. H. Vosko, L. Wilk, and M. Nusair. *Can. J. Phys*, 58:1200, 1980.

[15] C. Lee, W. Yang, and R. G. Parr. *Phys. Rev. B*, 57:785, 1988.

[16] B. Miehlich, A. Savin, H. Stoll, and H. Preuss. *Chem. Phys. Lett.*, 157:200, 1989.

[17] J. P. Perdew. *Phys. Rev. B*, 33:8822, 1986.

[18] J. P. Perdew, J. A. Chevary, S. H. Vosko, K. A. Jackson, M. R. Pederson, and C. Fiolhais. *Phys. Rev. B*, 46:6671, 1992.

[19] J. P. Perdew and A. Zunger. *Phys. Rev. B*, 23:5048, 1981.

[20] R. van Leeuwen and E. J. Baerends. Exchange-correlation potential with correct asymptotic behavior. *Phys. Rev. A*, 1994.

[21] A. D. Becke. *J. Chem. Phys.*, 98:5648, 1993.

[22] T. Yanai, D. P. Tew, and N. C. Handy. A new hybrid exchange-correlation functional using the Coulomb-attenuating method (cam-b3lyp). *Chem. Phys. Lett.*, 393:51, 2004.

[23] T. W. Keal D. J. Tozer and T. Helgaker. Giao shielding constants and indirect spin-spin coupling constants: performance of density functional methods. *Chem. Phys. Lett.*, 391:374, 2004.

[24] T. W. Keal and D. J. Tozer. A semi-empirical generalised gradient approximation exchange-correlation functional. *J. Chem. Phys.*, 121:5654, 2004.

[25] C. Adamo and V. Barone. *J. Chem. Phys.*, 110:6158, 1999.

[26] N. C. Handy C. W. Murray and G. J. Laming. *Mol. Phys.*, 78:997, 1993.

[27] S. Grimme. *J. Comp. Chem.*, 25:1463, 2004.

[28] S. Grimme. *J. Comp. Chem.*, 27:1787, 2006.

[29] P.-A. Malmqvist R. Lindh and L. Gagliardi. Molecular integrals by numerical quadrature. i. radial integration. *Theor. Chem. Acc.*, 106:178–187, 2001.

[30] O. Treutler and R. Ahlrichs. *J. Chem. Phys.*, 102:346, 1995.

[31] A. D. Becke. *J. Chem. Phys.*, 88:1053, 1988.

[32] G. Scuseria R. E. Stratmann and M. J. Frisch. *Chem. Phys. Lett.*, 257:213, 1996.

[33] P. Macak M. A. Watson, P. Salek and T. Helgaker. *J. Chem. Phys.*, 121:2915, 2004.

[34] S. Høst, B. Jansík, J. Olsen, P. Jørgensen, S. Reine, and T. Helgaker. A ground-state-directed optimization scheme for the Kohn—Sham energy. *Phys. Chem. Chem. Phys.*, 10:5344, 2008.

[35] S. Høst, J. Olsen, B. Jansík, L. Thøgersen, Poul Jørgensen, and T. Helgaker. The augmented Roothaan—Hall method for optimizing Hartree—Fock and Kohn—Sham density matrices. *J. Chem. Phys.*, 129:124106, 2008.

[36] M. Ziółkowski, V. Weijo, Poul Jørgensen, and J. Olsen. An efficient algorithm for solving nonlinear equations with a minimal number of trial vectors: Applications to atomic-orbital based coupled-cluster theory. *J. Chem. Phys.*, 128:204105, 2008.

[37] P. Pulay. *Chem. Phys. Lett.*, 73:393, 1980.

[38] P. Pulay. *J. Comp. Chem.*, 3:556, 1982.

[39] L. Thøgersen, J. Olsen, A. Köhn, P. Jørgensen, P. Sałek, and T. Helgaker. . *J. Chem. Phys.*, 123:074103, 2005.

[40] L. Thøgersen, J. Olsen, D. Yeager, P. Jørgensen, P. Sałek, and T. Helgaker. . *J. Chem. Phys.*, 121:16, 2004.

[41] K. N. Kudin, G. E. Scuseria, and E. Cancés. *J. Chem. Phys.*, 116:8255, 2002.

[42] Branislav Jansík, Stinne Høst, Poul Jørgensen, Jeppe Olsen, and Trygve Helgaker. Linear-scaling symmetric square-root decomposition of the overlap matrix. *J. Chem. Phys.*, 126:124104, 2007.

[43] B. Jansík, S. Høst, M. P. Johansson, J. Olsen, and P. Jørgensen. Robust and Reliable Multilevel Minimization of the Kohn—Sham Energy. *J. Chem. Theory Comput.*, 5:1027, 2009.

[44] B. Jansík, S. Høst, M. P. Johansson, J. Olsen, and P. Jørgensen. A stepwise atomic, valence–molecular, and full—molecular optimisation of the Hartree—Fock/Kohn—Sham energy. *Phys. Chem. Chem. Phys.*, 11:5805, 2009.

[45] P. Sałek, S. Høst, L. Thøgersen, P. Jørgensen, P. Manninen, Jeppe Olsen, B. Jansík, S. Reine, F. Pawłowski, E. Tellgren, T. Helgaker, and S. Coriani. Linear—scaling implementation of molecular electronic self—consistent field theory. *J. Chem. Phys.*, 126:114110, 2007.

[46] J. H. Van Lenthe, R. Zwaans, H. J. J. Van Dam, and M. F. Guest. Starting SCF calculations by superposition of atomic densities. *J. Comp. Chem.*, 27:926, 2006.

[47] V. Bakken and T. Helgaker. The efficient optimization of molecular geometries using redundant internal coordinates. *J. Chem. Phys.*, 117:9160, 2002.

[48] R. Lindh, A. Bernhardsson, G. Karlström, and P.-Å. Malmqvist. On the use of a hessian model function in molecular geometry optimizations. *Chem. Phys. Lett.*, 241:423, 1995.

[49] S. Reine, A. Krapp, M. F. Iozzi, V. Bakken, T. Helgaker, F. Pawłowski, and P. Sałek. An efficient density-functional-theory force evaluation for large molecular systems. *J. Chem. Phys.*, 133:044102, 2010.

[50] J. Baker. Techniques for geometry optimization: A comparison of cartesian and natural internal coordinates. *J. Comp. Chem.*, 14:1085, 1993.

Part IV

Index